THE UNIVERSITY OF ALBERTA

A HYBRID COMPUTER TECHNIQUE FOR SOME OPTIMAL CONTROL PROBLEMS

by

(C) GEOFFREY CHARLES MICHAELS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

SPRING, 1970

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "A Hybrid Computer Technique for Some Optimal Control Problems", submitted by Geoffrey C. Michaels in partial fulfilment of the requirements for the degree of Master of Science.

ABSTRACT

The theory of optimal control of finite-dimensional differential dynamic systems has been extensively developed. The application of Pontryagin's Minimum Principle often leaves only a two-point boundary-value problem for solution. The solution of these problems is an area of expanding concern as application of optimal control is attempted. Digital computer methods have been developed. In this thesis, the hybrid computer is used in solution of these problems.

A recent hybrid computer technique has been extended to a more general target set case. The underlying assumptions have been shown to be inapplicable in general, thus limiting the class of problems for which the technique is valid. A program using the hybrid computer for applying the technique to second order systems was written and tested.

ACKNOWLEDGEMENTS

The author would like to express his appreciation to Dr. V. Gourishankar for his encouragement and guidance in preparing this work, to Dr. R. Rink for some useful discussions of the derivations, and to P. Harding for some assistance in the hybrid computer work.

The financial assistance directly provided to the author by Calgary Power Ltd., through the G.A. Gaherty Memorial Scholarship, and by the estate of Mrs. Dorothy J. Killam, through the Izaak Walton Killam Memorial Fellowship is gratefully acknowledged. The financial assistance of the National Research Council and of the department is acknowledged in supporting the hybrid computer facility used.

TABLE OF CONTENTS

MAJOR SYMBOLS

Symbols

C	Cost
D	Difference between target set and state
E	Terminal error
f	State derivative
G	Target set
H	Hamiltonian function
J	Cost functional
k	Positive constant
m	Control space dimension
n	State space dimension
p	Costate
R	Reachable set
t	Time
T	Terminal time
u	Control function
U	Admissible set of controls
x	State

Superscripts

(o)	Initial quantity
*	Optimal quantity
+	Arbitrary quantity
T	Transpose
·	Time derivative
–	Closure of a set
^	Augmented quantity

Subscripts

o	Initial quantity
f	Final quantity
T	Terminal quantity
0,1,...n	Coordinate
–	Vector or matrix

LIST OF FIGURES AND TABLES

CHAPTER I

INTRODUCTION

1.1 The Optimal Control Problem

In recent years, several approaches have been developed for the automatic control of physical systems in an optimal manner. A hybrid computer technique for some optimal control problems is presented in this thesis. The systems to which this technique is applicable fall in the class of finite-dimensional continuous-time differential dynamical systems[1]*. See Appendix 1 for definitions. It is assumed that the systems can be simulated on an analog computer.

The statement of the optimal control problem used in this thesis follows. The plant or process is represented by a vector-matrix differential equation called the state equation

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}, t). \tag{1.1}$$

The block diagram of Figure 1.1 represents the system.

The problem is defined over some connected interval of time, I, with

$$t \in I. \tag{1.2}$$

The state, $\underline{x}(t)$, represents the changing characteristics of the system, given by the solution to (1.1). With E^n taken as Euclidean n-space,

$$\underline{x}(t) \in E^n. \tag{1.3}$$

No other constraints are placed on $\underline{x}(t)$.

---

* Numbers in square brackets refer to the literature cited in the Bibliography.

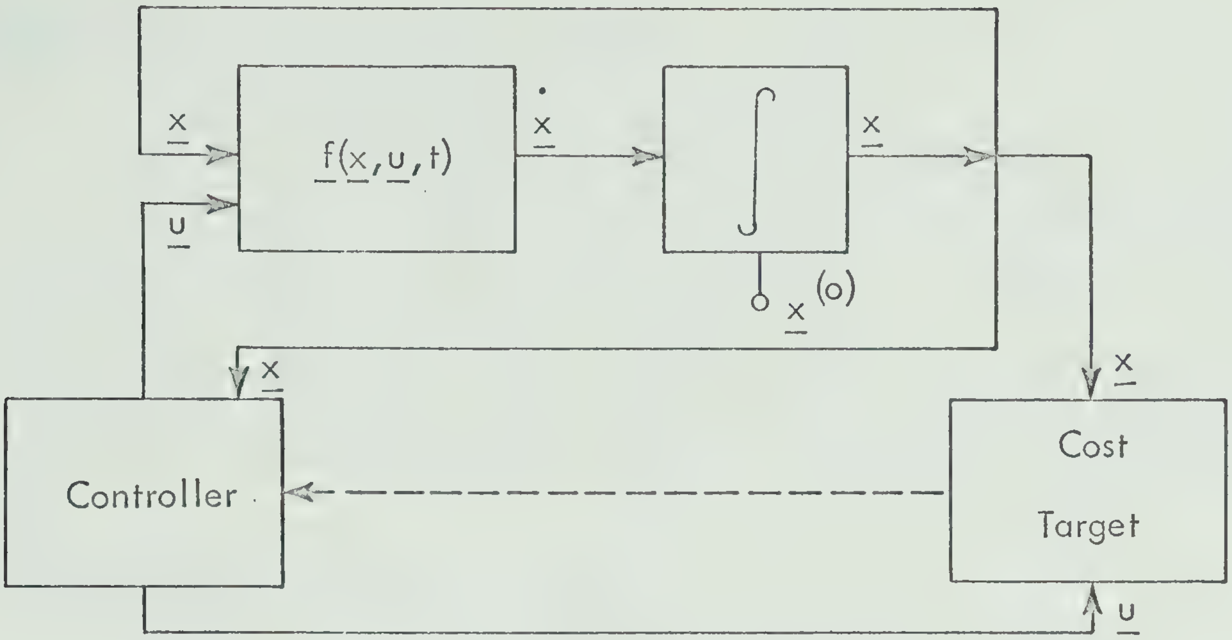


FIGURE 1.1 BLOCK DIAGRAM OF THE SYSTEM

The control function, $\underline{u}(t)$, represents the input to the system by which the output is determined.

$$\underline{u}(t) \in U \subset E^m. \tag{1.4}$$

It is assumed that $\underline{u}(t)$ is not subject to any state-dependent constraints.

The index of performance or cost functional of the system is taken to be the integral of some scalar function of the states, controls and time.

$$J(t) = \int_{t_o}^{t} f_o(\underline{x}, \underline{u}, \tau)\, d\tau \tag{1.5}$$

where

$$t \in [t_o, t_f] \subset I . \tag{1.6}$$

The state of the system is initially given by

$$\underline{x}(t_o) = \underline{x}^{(o)}(t_o). \tag{1.7}$$

The final state is required to be

$$\underline{x}(t_f) \in G(\underline{x}, t) . \tag{1.8}$$

where

$$G(\underline{x}, t) = \{\underline{x}(t) \in E^n | \underline{g}(\underline{x}, t) \leq 0\}. \quad (1.9)$$

The optimal control problem involves the determination of the optimal control $u^*(t)$ from among the class of admissible controls given by (1.4) which transfers the state $x(t)$ of (1.1) from the initial state (1.7) to the target set (1.8) with the minimum value of cost (1.5).

1.2 Pontryagin's Minimum Principle

The use of Pontryagin's minimum principle is one of several possible approaches to solution of the problem posed in the previous section. The use of this principle yields the necessary conditions which an optimal control function must satisfy. These conditions are also sufficient conditions for all extremal controls. Several excellent textbooks containing a detailed derivation of these conditions are available. Some of these are noted in the bibliography[1,8,13]. It is assumed in this thesis that the minimum principle can be applied to the systems studied.

The relevant parts of the necessary conditions are briefly stated here.

Define the costate vector, $\underline{p}(t)$, a form of Lagrange multiplier,

$$\underline{p}(t) \in E^n. \quad (1.10)$$

Augment the costate vector with $p_o(t)$.

$$\hat{\underline{p}}(t) = (p_o(t), \underline{p}(t))^T \in E^{n+1} \quad (1.11)$$

where $(\cdot)^T$ means the transpose.

Augment the state vector with J, by letting

$$x_o(t) = J(t); \; x_o(t_o) = 0 \quad (1.12)$$

so that $\dot{\hat{\underline{x}}} = \hat{\underline{f}}(\hat{\underline{x}}, \underline{u}, t) = \hat{\underline{f}}(\underline{x}, \underline{u}, t)$

$$= (f_o(\underline{x}, \underline{u}, t), \quad \underline{f}(\underline{x}, \underline{u}, t))^T . \tag{1.13}$$

Define the Hamiltonian function as

$$H(\hat{\underline{x}}, \underline{u}, t, \hat{\underline{p}}) = \hat{\underline{p}} \cdot \hat{\underline{f}}. \tag{1.14}$$

Clearly

$$\dot{\hat{\underline{x}}}(t) = \frac{\partial H}{\partial \hat{\underline{p}}} . \tag{1.15}$$

To specify $\hat{\underline{p}}(t)$, we define

$$\dot{\hat{\underline{p}}}(t) = - \frac{\partial H}{\partial \hat{\underline{x}}} . \tag{1.16}$$

One of the necessary conditions required by the minimum principle is that the Hamiltonian for an extremal control be minimum with respect to all other controls. That is, for $u^*(t)$ an extremal control

$$H(\hat{\underline{x}}, \underline{u}^*, t, \hat{\underline{p}}) \leq H(\hat{\underline{x}}, \underline{u}, t, \hat{\underline{p}}) . \tag{1.17}$$

This condition will ordinarily determine the extremal controls as a function of the states and costates, found by solving (1.15) and (1.16).

$$\underline{u}(t) = \underline{v}(\hat{\underline{x}}, \hat{\underline{p}}, t) . \tag{1.18}$$

Using these necessary conditions, extremal trajectories can be generated. The problem is reduced to what is termed the two-point boundary-value problem, finding the correct initial conditions for the state and costate which bring the extremal trajectory to the specified final conditions. An extremal trajectory thus found may then be tested for optimality.

1.3 Solution of the Two-Point Boundary-Value Problem

It is clear from the previous section that the solution of the two-point boundary-value problem is essential for the application of the minimum principle. Several methods have been proposed

for solving the two-point boundary-value problem.

For the simplest of problems, analytical solutions may be found. Athans and Falb[1] develop these techniques for a few systems. In most cases, however, some form of iterative procedure must be used.

One strategy would be to guess a set of unknown initial conditions, solve the system of differential equations for those conditions, and correct the initial conditions so as to better match the final conditions. Thus, this strategy is a parameter search on the missing initial conditions carried out by the iterative solution of a set of initial condition problems. Many other strategies exist, and there are many ways of implementing each. Bekey and Karplus[3], Balakrishnan and Neustadt[2], and McLeod[9] provide starting points for literature surveys. The strategy described here is widely used and will be used in this thesis in a modified form.

Solution may be attempted by pure analog methods, with human decision-making. Using repetitive operation, trial and error searches are feasible on up to third order systems, at least. Some of the articles in McLeod[9] examine analog methods applied to two-point boundary-value problems. Analog computation is severely disadvantaged by the inability of analog computation to handle anything but the simplest logical decisions.

The usual tool for solving two-point boundary-value problems is the digital computer. One of many possible sources, Balakrishnan and Neustadt[2] gives a survey of the application of a number of techniques.

Some general concepts of digital computer solution of two-point boundary-value problems using the general strategy described

are presented here.

The accuracy of digital computer solutions can be very high. In cases where the accuracy requirements on the solution are higher than the .1% to 1% possible with modern analog computers, there is no alternative to using a digital computer. Such requirements arise, for example, in satellite trajectory computations. Often, however, the accuracy is a by-product of the necessity for preventing accumulated round-off errors from destroying all accuracy of a digital computer solution, and is not required (or usable) in many engineering applications. Appendix 2 expands these concepts.

In terms of computer utilization, digital computer solutions are often expensive in that they take a large amount of storage and much computation time. Specifically, a fast computer with a large amount of storage is required if a problem is to be solved quickly in real time. Solution on a "small" digital computer will in general be slow in real time and in some cases may be impossible because of lack of storage capability. This situation is caused by two factors in the solution of differential equations by the use of a digital computer. The system must be solved as a set of difference equations, which means that a large number of calculations must be carried out to high accuracy in order to assure an accurate solution and to avoid cumulative round-off and systematic errors which may otherwise occur. Each calculation by a digital computer must be accomplished serially, increasing solution time with system complexity. Use of the analog computer for solution of differential equations does not have these disadvantages.

Hybrid computation has the advantage that the best points of analog and digital computation can be used. The speed of a high speed

analog computer can be used for the solution of the differential equation system. The logical, storage and calculation capabilities of the digital computer can be used for implementation of automatic iteration schemes. The accuracy possible is limited by the analog solution. Further features of hybrid computation are discussed in Appendix 2. In many engineering situations, a fast and inexpensive solution tool is available to do a job which could be done equally well only by a much more expensive digital computer.

1.4 Scope of the Thesis

A computer program for the solution of some optimal control problems is presented. Pontryagin's minimum principle is used to determine a two-point boundary-value problem. A hybrid computer available in the Electrical Engineering Department is used to show that the computational technique is fast compared to some digital computer techniques. Appendix 3 gives a brief description of the hybrid computer used.

The technique presented here was inspired by a paper by T. Miura, J. Tsuda and J. Iwata[10]. The authors presented their approach for point target sets at the origin. They have given no details of their computational methods.

This thesis presents a derivation of an extension of their work, an improvement in technique of application of the derived conditions, and a limitation of the class of problems for which the technique is applicable.

The derivation is extended to the case of general fixed target sets, with the method of attack for time-varying target sets indicated but not derived. The algorithm derived by Miura et al turns

out to be a special case of the general algorithm.

The technique of application is improved by the use of high speed analog computer techniques to quickly solve the differential equations; by fast digital computation and control by the use of a low-level assembler language; and by the use of a steepest descent search scheme to bring rapid convergence[5].

The class of problems for which the technique converges to the optimal solution is restricted and it is shown that the derivation in Miura et al[10] is incorrect as given.

The derivation of the algorithm is given in Chapter 2. The method of solution and the hybrid computer program are discussed in Chapter 3. The application of the technique to two second order systems is discussed in Chapter 4. The fifth and last chapter of the thesis consists of a discussion of the results and suggestions for future work. Four appendices are provided for related reference.

CHAPTER 2

DEVELOPMENT OF THE TECHNIQUE

2.1 Introduction

The technique used to reach the optimal control solution is described in this chapter. This technique is basically a method of solving the two-point boundary-value problem which results when Pontryagin's minimum principle is applied to the control problem. The conditions for convergence of this technique are examined. Both fixed and time-varying target sets are considered. A detailed derivation is included for the case of fixed target sets, whereas no derivation is given for the technique applied to the time-varying case.

2.2 Fixed Target Sets

For each of a monotonically increasing set of costs, the terminal state is brought as close as possible to the target set. This process is terminated when the terminal state is sufficiently close to or at the target set. The value of initial costate which brings this solution is defined as the solution of the two-point boundary-value problem.

Statement of the Problem

Given the dynamical system of (1.13) with

$$\dot{\hat{\underline{x}}} = \hat{\underline{f}}(\hat{\underline{x}}, \underline{u}, t) \tag{2.1}$$

with initial state

$$\underline{x}^{(o)}(t_o) \tag{2.2}$$

and final state in the target set given by (1.9)

$$\hat{\underline{x}}(t_f) \in G\ (\hat{\underline{x}}) \tag{2.3}$$

and controls

$$\underline{u}(t) = \underline{v}(\hat{\underline{x}}, \hat{\underline{p}}) \tag{2.4}$$

from (1.18), find the set of initial conditions, $\underline{\eta}^*$, of the costate equation (1.16)

$$\dot{\hat{\underline{p}}}(t) = -\frac{\partial H}{\partial \hat{\underline{x}}} \tag{2.5}$$

such that (2.1), (2.2), (2.3), (2.4) and (2.5) are all simultaneously satisfied. The solution vector, $\underline{\eta}^*$, is the solution of the two-point boundary-value problem.

It is assumed that there exists a solution $\underline{\eta}^*$ which solves the problem in a finite time, t_f^*, and with a finite minimum cost, C^*.

Before proceeding with the derivation of the technique, a few definitions will be given.

The Banach Space E^n

With the Euclidean norm, $||\underline{x}|| = (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}}$, $\underline{x} \in E^n$, and the norm-derived metric $\rho(\underline{x}, \underline{y}) = ||\underline{x} - \underline{y}||$; $\underline{x}, \underline{y} \in E^n$, ρ being the distance function; the space E^n is a Banach space[12]. That is, it is a normed (thus metric), linear, and complete space of vectors.

Distance of a vector from a set

The distance from a vector $\underline{x}$ to a set G is defined as

$\rho(\underline{x}, G) = \inf_{\underline{y} \in G} \rho(\underline{x}, \underline{y})$.

Vector from a vector to a set

The vectors from a vector $\underline{x}$ to a set G are

$G - \underline{x} = \{\underline{y} - \underline{x} \mid \rho(\underline{x}, \underline{y}) = \rho(\underline{x}, G), \underline{y} \in \overline{G}\}$, which may not be unique but which have unique length. These last two concepts are illustrated in Figure 2.1.

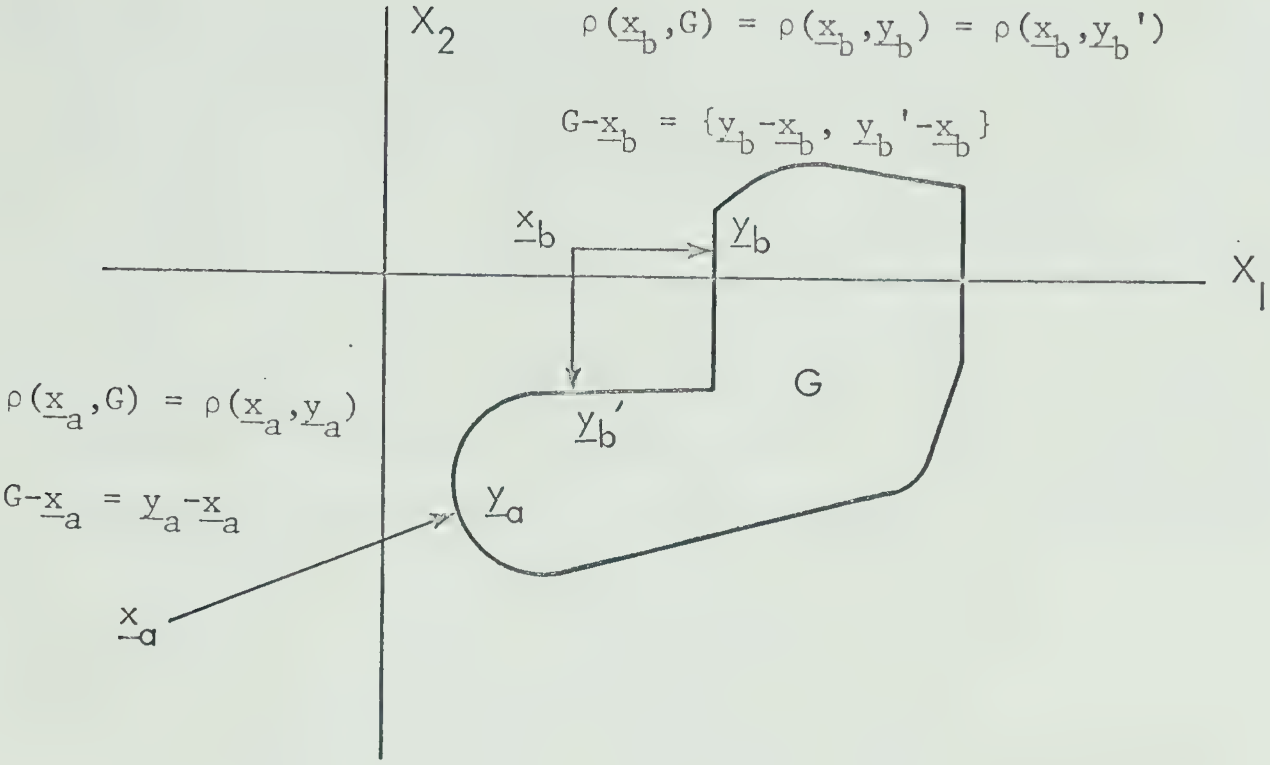


FIGURE 2.1 SOME DISTANCE NOTIONS IN E^2

The reachable set, called R(C), is defined as the set of all states which may be reached from $\underline{x}^{(o)}(t_o)$ in cost C using the extremal control strategy. In symbolic notation this can be stated as

$$R(C) = \{\hat{\underline{x}}\,(T) \mid \hat{\underline{x}}\,(T) \in E^{n+1}\ ;\quad J = C \quad \text{at} \quad t = T;$$
$$\text{and (2.1), (2.2), (2.4), (2.5) hold }\}. \tag{2.6}$$

Clearly $R(C) \cap G\ (\hat{\underline{x}}) \neq \phi$ if and only if $C \geq C^*$. Consider $R(C) \cap G\ (\hat{\underline{x}}) \neq \phi$, which implies that there exists a vector, $\underline{z}(T)$, such that $\underline{z}(T) \in R(C)$ and $\underline{z}(T) \in G\ (\hat{\underline{x}})$. But then (2.3) is satisfied as well as the others given in (2.6), the definition of R(C), and the problem is solved. Thus $C = C^*$ since this is the minimum extremal solution. The reverse case is obvious. Thus $R(C) \cap G\ (\hat{\underline{x}})$ is nonempty first at $C = C^*$. The approach used is based on this observation.

Derivation of the Method

The underlying principle of the method is outlined here.

Beginning with a cost of zero and using an increasing set of costs, the terminal state (in the reachable set) is kept as close as possible to the target set. If this condition is maintained from C = 0 until C = C*, then the initial costate corresponding to the last iteration which had C = C* and which brought the terminal state to the target set is the one which produces the optimal control, because $R(C^*) \cap G(\underline{\hat{x}}) = \underline{\hat{x}}(t_f)$ and equations (2.1) to (2.5) are all satisfied.

Formulation of an Equivalent Problem

Consider operating the system* with a particular cost C and an initial costate $\underline{\eta}$. The terminal states and costates can be considered to be functions of the parameter C and the vector parameter $\underline{\eta}$, since fixing these values and operating the system must give a unique result. The terminal state and costate are thus written as $\underline{\hat{x}}_T(C, \underline{\eta})$ and $\underline{\hat{p}}_T(C, \underline{\eta})$. Since $G(\underline{\hat{x}})$ is a set in E^{n+1} independent of C and $\underline{\eta}$, G can be written for $G(\underline{\hat{x}})$. Then the vector $G(\underline{\hat{x}}) - \underline{x}(C, \underline{\eta})$ can be written $G - \underline{x}(C, \underline{\eta}) = \underline{D}(C, \underline{\eta}) \varepsilon E^{n+1}$. Of course, $||\underline{D}(C, \underline{\eta})|| = ||G - \underline{\hat{x}}(C, \underline{\eta})|| > 0$ for $C < C^*$.

The problem to be solved is that of finding the correct value of initial costate for any cost C. Consider any value of $\underline{\eta}$, say $\underline{\eta}^+$. At t^+, J = C and $\underline{D}^+ = \underline{D}(C, \underline{\eta}^+)$. $\underline{D}^+$ is a fixed vector in E^{n+1}. The new objective is to minimize the cost functional

$$J' = k(\underline{D}^+, \underline{D}(C, \underline{\eta})) \tag{2.7}$$

---

* The phrase "operating the system" is taken to mean the following. For a given problem, assign a value to $\underline{\eta}$, the initial costate and solve the system equations until the cost is C. The time at which this occurs is called the terminal time, T. The state and costate are the terminal state and costate, $\underline{\hat{x}}(T)$ and $\underline{\hat{p}}(T)$, respectively.

where the inner product used is $(\underline{a}, \underline{b}) = \underline{a}^T \underline{b}$, $\underline{a}, \underline{b} \in E^n$.

There is at least one $\underline{y} \in G$ such that

$$G - \hat{\underline{x}}(C, \underline{n}) = \underline{y} - \hat{\underline{x}}(C, \underline{n}) = \underline{D}(C, \underline{n}). \tag{2.8}$$

These values of $\underline{y}$ (if not unique) are functions of the state $\hat{\underline{x}}$, and all give rise to the same value of $||\underline{D}||$.

Constraints

There are two possible characteristics of G for any $\underline{y}$. The tangent plane may be unique, in which case $\underline{D}(C, \underline{n})$ is normal to all vectors in this tangent plane. The tangent plane to G may not exist at $\underline{y}$, in which case, if one considers a hypersphere around $\underline{y}$, $\underline{D}(C, \underline{n})$ can be considered as normal to the hypersphere for any non-zero radius. It will be a computational necessity to have a small tolerance around such a point, thus the use of a tangent plane to a hypersphere around the point is a useful concept. Figure 2.2 illustrates these concepts in E^3.

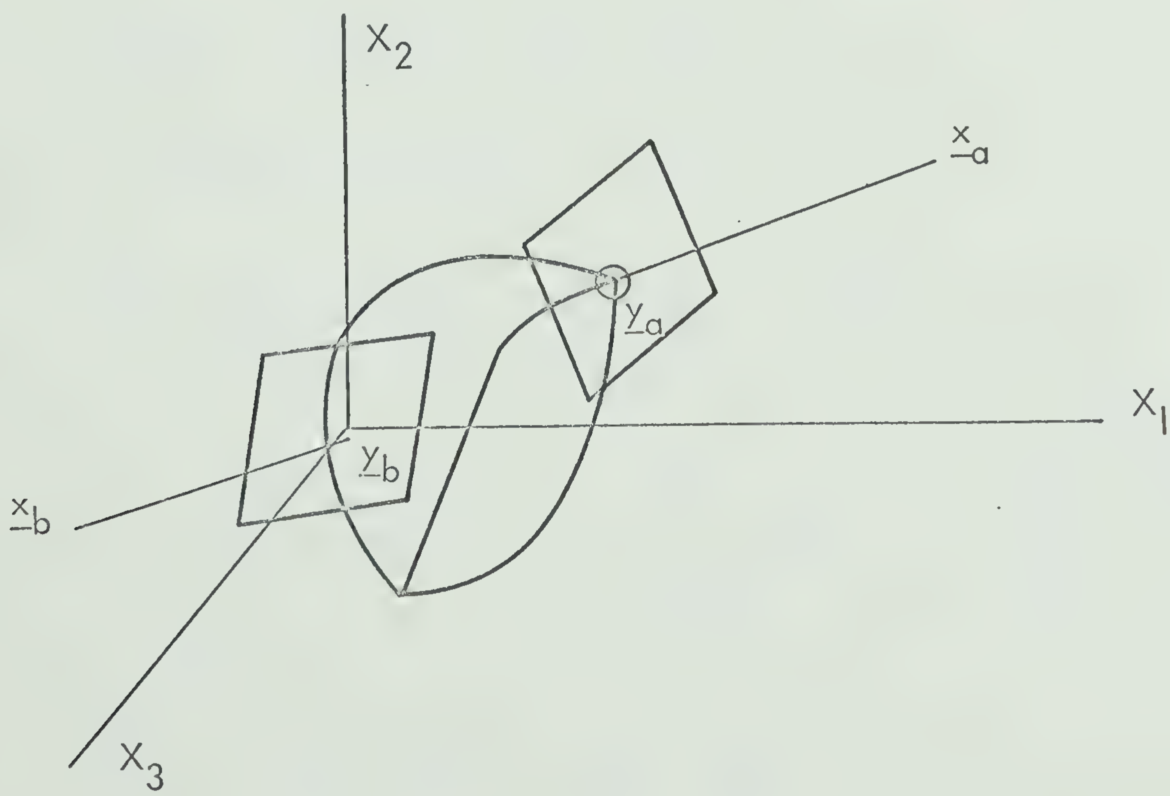


FIGURE 2.2 TANGENT HYPERPLANES IN E^3

Restrictions are placed on the state space and the target set.

In general the cost is unconstrained so that

$$G(\hat{\underline{x}}) = G(\underline{x}) . \tag{2.9}$$

It is assumed that the space E^n is separated into two sets G and $H = G^c$ (= $\{\underline{x} | \underline{x} \varepsilon E^n$ and $x \notin G\}$). $H = \bigcup_{\alpha \varepsilon A} H_\alpha$, with each H_α being characterized by a specific value of $\underline{x}_\alpha$ and a constant k_α, and with A a set of indices, quite possibly nondenumerable. For $\underline{x} \varepsilon H_\alpha$,

$$\underline{D} = \frac{(\underline{x} - \underline{x}_\alpha)}{||\underline{x} - \underline{x}_\alpha||} (||\underline{x} - \underline{x}_\alpha|| - k_\alpha) , \underline{x} \varepsilon H_\alpha . \tag{2.10}$$

When $$\underline{x} \varepsilon H_{\alpha 1} \cap H_{\alpha 2} \tag{2.11}$$

$$\underline{D} = \frac{(\underline{x} - \underline{x}_{\alpha 1})}{||\underline{x} - \underline{x}_{\alpha 1}||} (||\underline{x} - \underline{x}_{\alpha 1}|| - k_{\alpha 1})$$

$$= \frac{(\underline{x} - \underline{x}_{\alpha 2})}{||\underline{x} - \underline{x}_{\alpha 2}||} (||\underline{x} - \underline{x}_{\alpha 2}|| - k_{\alpha 2}) . \tag{2.12}$$

This form of $\underline{D}$ means that the following equality holds.

$$\frac{\underline{D}}{||\underline{D}||} = \frac{G - \underline{x}}{||G - \underline{x}||} = \frac{\underline{x}_\alpha - H_\alpha}{||\underline{x}_\alpha - H_\alpha||} , \forall \underline{x} \varepsilon H_\alpha . \tag{2.13}$$

$\underline{x}_\alpha$ can be considered to be the "attracting point" for $\underline{D}$ corresponding to each $\underline{x}$.

In order to analyse the system on a computer, the sets H_α must be grouped into a finite collection H_β, $\beta \varepsilon B$, with $\bigcup_{\beta \varepsilon B} H_\beta = \bigcup_{\alpha \varepsilon A} H_\alpha = G^c$. The unifying feature of each H_β is the function which determines the set of attracting points $\underline{x}_\alpha$ for all $H_\alpha \varepsilon H_\beta$. For example, if a set of $\underline{x}_\alpha$ form a line segment, then H_β would be the domain of state space using that line segment as a set of attracting points, and the line segment (and thus each $\underline{x}_\alpha$) would be given by a particular function

of $\underline{x}$, for $\underline{x} \varepsilon H_\beta$. These concepts are demonstrated in the following examples and in Figures 2.3 to 2.6.

Examples

1. Hyperspheres - Figure 2.4

$$G = \{\underline{g} \mid ||\underline{g} - \underline{g}_o|| \leq R \ ; \ \underline{g}, \underline{g}_o \ \varepsilon \ E^n\} \tag{2.14}$$

$$\underline{D} = \frac{(\underline{g}_o - \underline{x})}{(||\underline{g}_o - \underline{x}||)} (||\underline{g}_o - \underline{x}|| - R) \qquad \underline{x} \ \varepsilon \ G^c. \tag{2.15}$$

In the framework of the derivation, only one region of $H = G^c$ need be considered, with

$$\underline{x}_\alpha = \underline{x}_\beta = \underline{g}_o, \ k_\alpha = k_\beta = R \tag{2.16}$$

for a single α and β, $\underline{x} \ \varepsilon \ H_\alpha = H_\beta \ \forall \underline{x} \ \varepsilon \ G^c$.

The work of Miura, et al[10] is concerned with a special case of this example. The technique is derived there for the target set being the origin of state space. In their paper, then, they have used $\underline{g}_o = \underline{0}$ and $R = 0$, so that $\underline{D} = -\underline{x}$. Their result is

$$\underline{p}(\tau) = - k\underline{x}(\tau) \tag{2.17}$$

which is the same as given by (2.35) in this case,

$$\underline{p}_T(C, \underline{n}) = - k\underline{x}(C, \underline{n}). \tag{2.18}$$

2. Infinite Hypercylinders - Figure 2.5

$$g_\ell = \{\underline{g} \mid \underline{g} = \alpha \underline{g}_1 + (1 - \alpha) \ \underline{g}_2 \ ; \ \underline{g}_1, \underline{g}_2, \underline{g} \ \varepsilon \ E^n;$$
$$\alpha \ \varepsilon \ (-\infty, \infty)\}. \tag{2.19}$$

a line in state space through $\underline{g}_1$ and $\underline{g}_2$.

$$G = \{\underline{g} \mid ||\underline{g} - g_\ell|| \leq R; \ \underline{g} \ \varepsilon \ E^n. \tag{2.20}$$

Here $\quad H_\alpha = \{\underline{x} \mid (\underline{x} - g_\ell) = (\underline{x} - \underline{g}_{\ell\alpha}); \ \underline{g}_{\ell\alpha} \ \varepsilon \ g_\ell$ for
$$\alpha \varepsilon \ (-\infty, \infty); \ \underline{x} \ \varepsilon \ H\}. \tag{2.21}$$

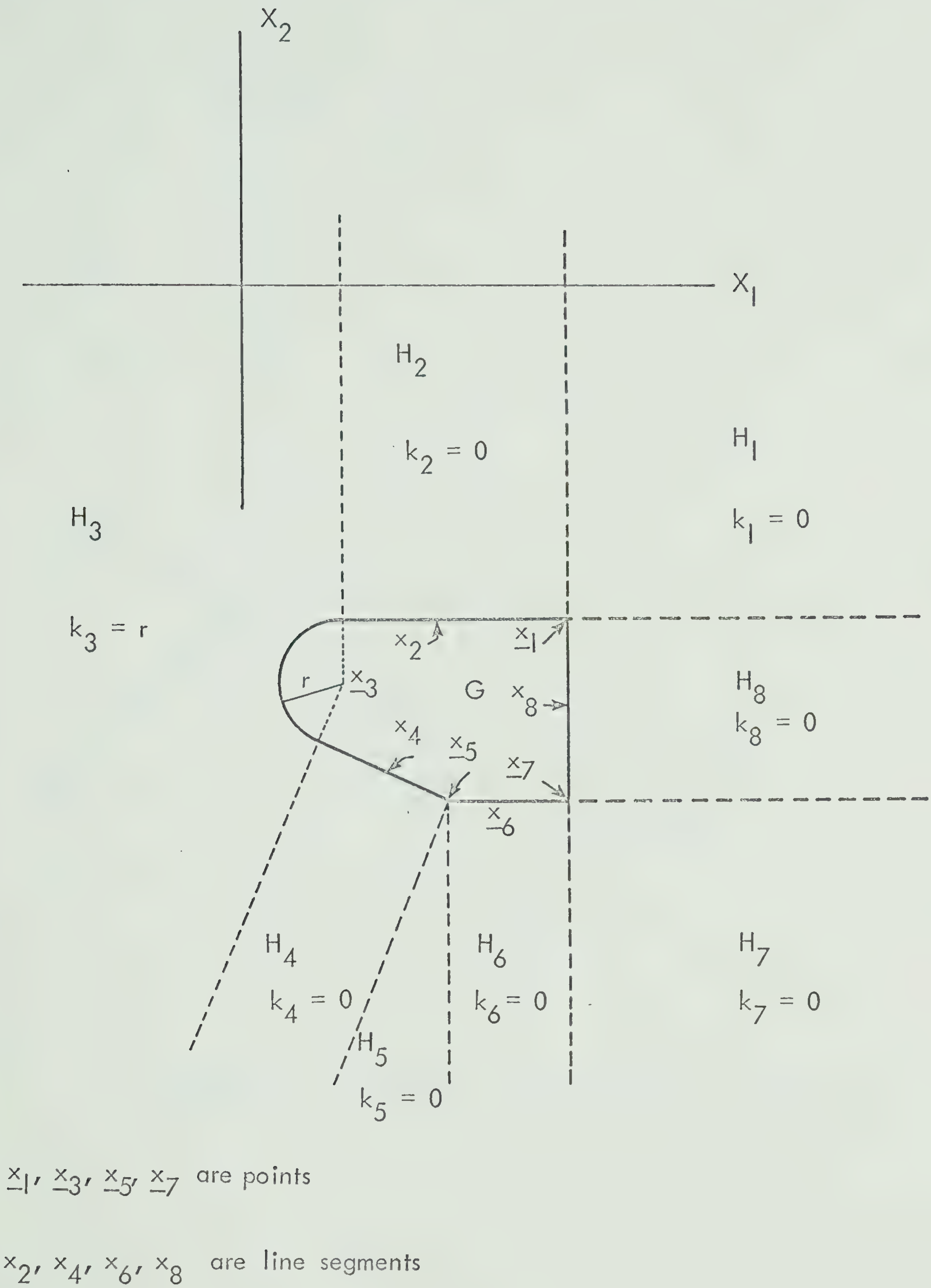


FIGURE 2.3 ATTRACTION FUNCTIONS IN STATE SPACE

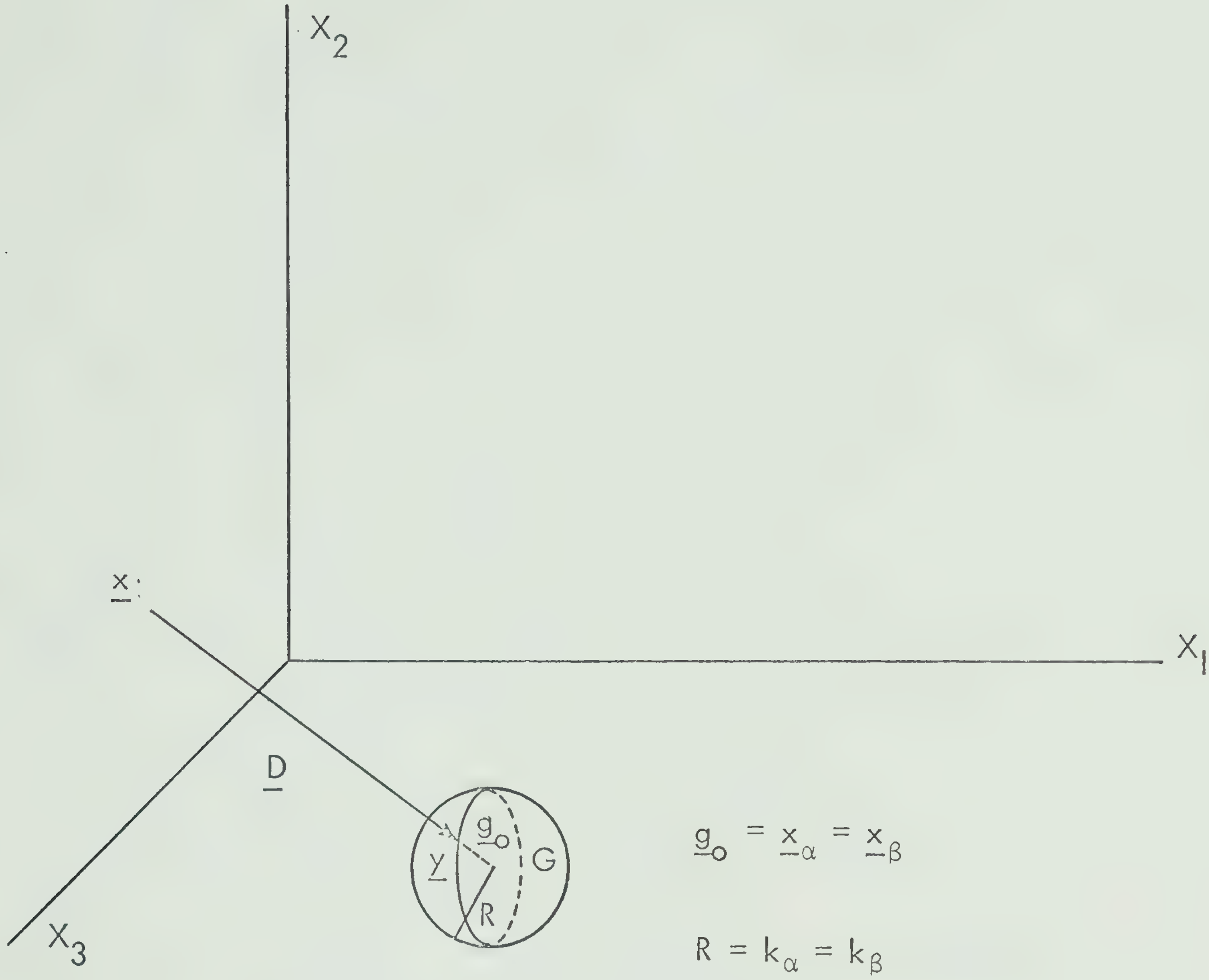


FIGURE 2.4 EXAMPLE I: HYPERSPHERICAL TARGET SET

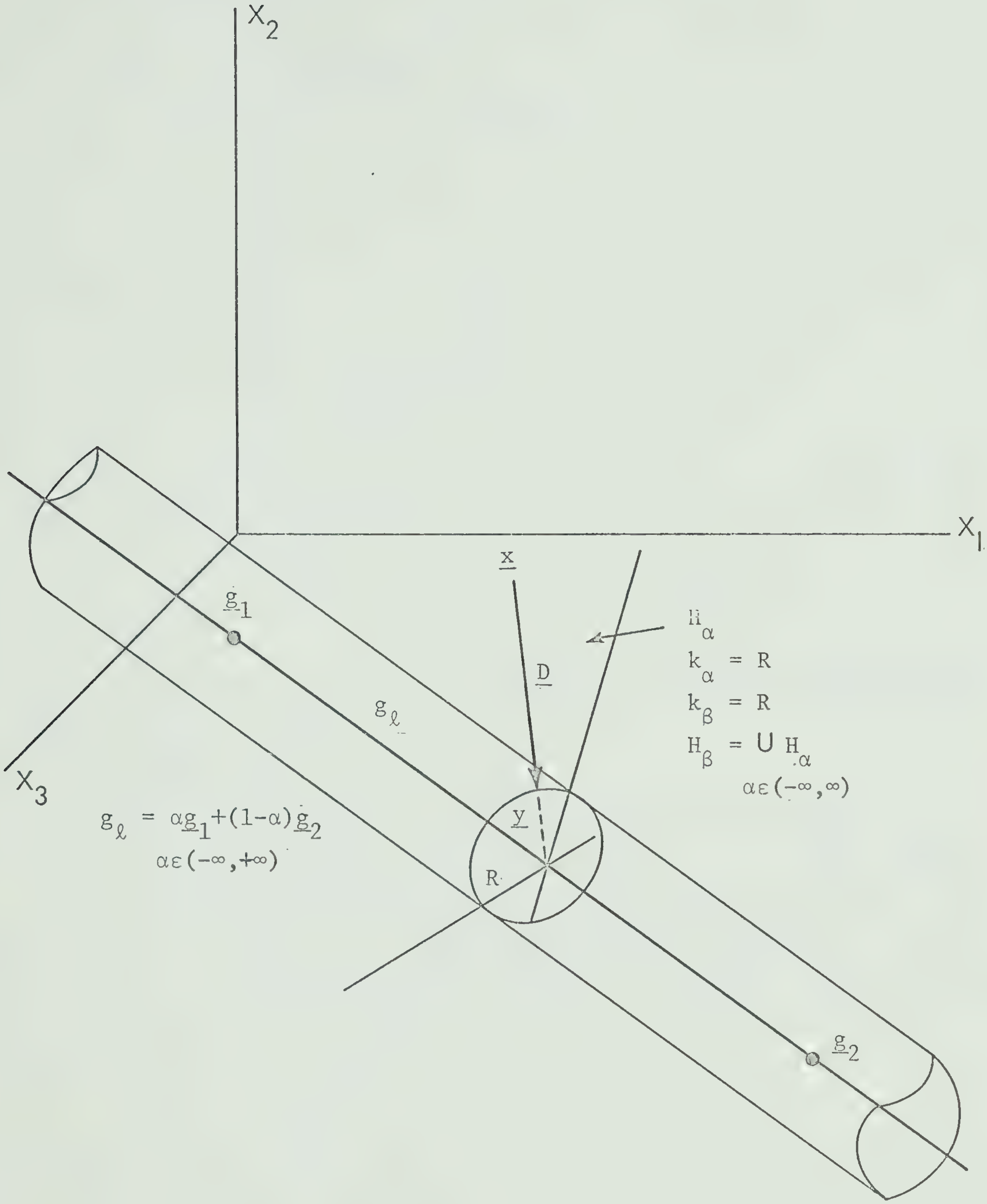


FIGURE 2.5a EXAMPLE 2: INFINITE HYPERCYLINDRICAL TARGET SET

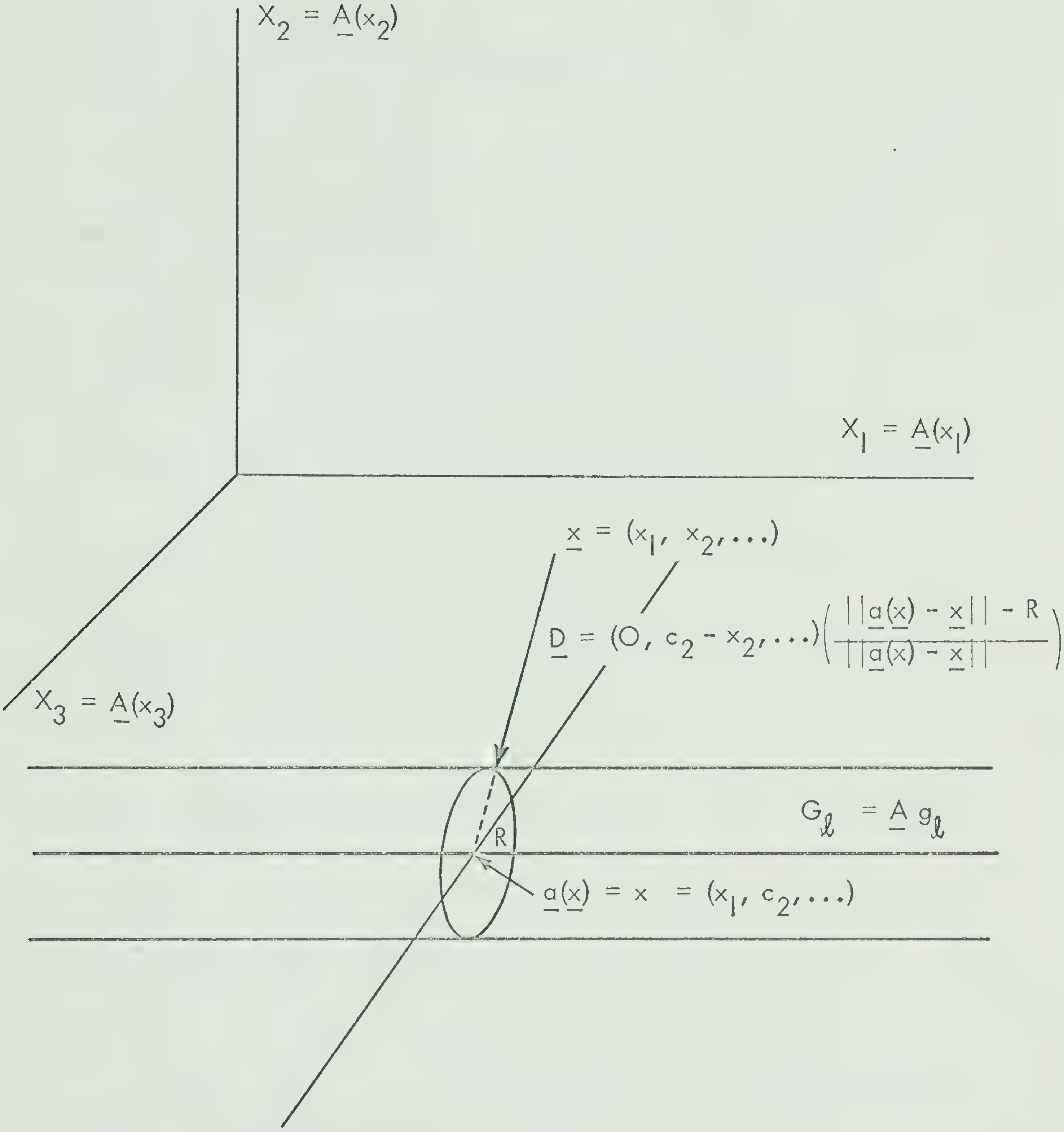


FIGURE 2.5b EXAMPLE 2: INFINITE HYPERCYLINDRICAL TARGET SET IN TRANSFORMED STATE SPACE

$$\underline{x}_\alpha = \underline{g}_{\ell\alpha} = \alpha\underline{g}_1 + (1 - \alpha)\ \underline{g}_2 \quad k_\alpha = R$$

$$H_\beta = \cup H_\alpha \qquad \alpha\varepsilon\ (-\infty, \infty)$$

x_β is the set of points forming g_ℓ

$$k_\beta = R \quad \text{for a single } \beta \tag{2.22}$$

Here again only one region need be considered with g_ℓ being the attracting function (representing a set of attracting points).

Computational considerations indicate that an effective method for treating this type of target set would be by transformation of the system equations by the linear transformation matrix $\underline{A}$ with unity norm which takes the set g_ℓ into a line in (say) the x_1 direction.

$$\underline{A}g_\ell = G_\ell = \{\underline{x} | \underline{x} = (\alpha, c_2, c_3, \ldots, c_n);\ \alpha\varepsilon\ E^1;\ c_2 \text{ to } c_n \text{ are real constants}\}. \tag{2.23}$$

Assuming that the system equations have been so transformed, we have

$$\underline{a}(\underline{x}) = (x_1, c_2, c_3, \ldots, c_n) \tag{2.24}$$

and

$$\underline{D} = \frac{(\underline{a}(x) - \underline{x})}{||\underline{a}(x) - \underline{x}||}\ (||\underline{a}(\underline{x}) - \underline{x}|| - R). \tag{2.25}$$

Note that $\underline{a}(\underline{x}) - \underline{x} = (0, c_2 - x_2, c_3 - x_3, \ldots, c_n - x_n)$. (2.26)

This makes the calculation of $\underline{D}$ convenient in practice.

3. Hypercylinders with Hyperhemispherical Ends - Figure 2.6

$$g_\ell = \{\underline{g} | \underline{g} = \alpha\underline{g}_1 + (1 - \alpha)\ \underline{g}_2;\ \underline{g}_1, \underline{g}_2, \underline{g}\ \varepsilon\ E^n;\ \alpha\varepsilon[0,1]\} \tag{2.27}$$

a line segment in state space from $\underline{g}_1$ to $\underline{g}_2$.

$$G = \{\underline{g}|\ ||\underline{g} - g_\ell|| \leq R;\ \underline{g}\ \varepsilon\ E^n\}. \tag{2.28}$$

Consider a transformation $\underline{A}$ as in the previous example. Then

$$\begin{aligned} \underline{b}_1 &= \underline{A}\ \underline{g}_1 = (\alpha_1, c_2, c_3, \ldots, c_n) \\ \underline{b}_2 &= \underline{A}\ \underline{g}_2 = (\alpha_2, c_2, c_3, \ldots, c_n) \end{aligned} \tag{2.29}$$

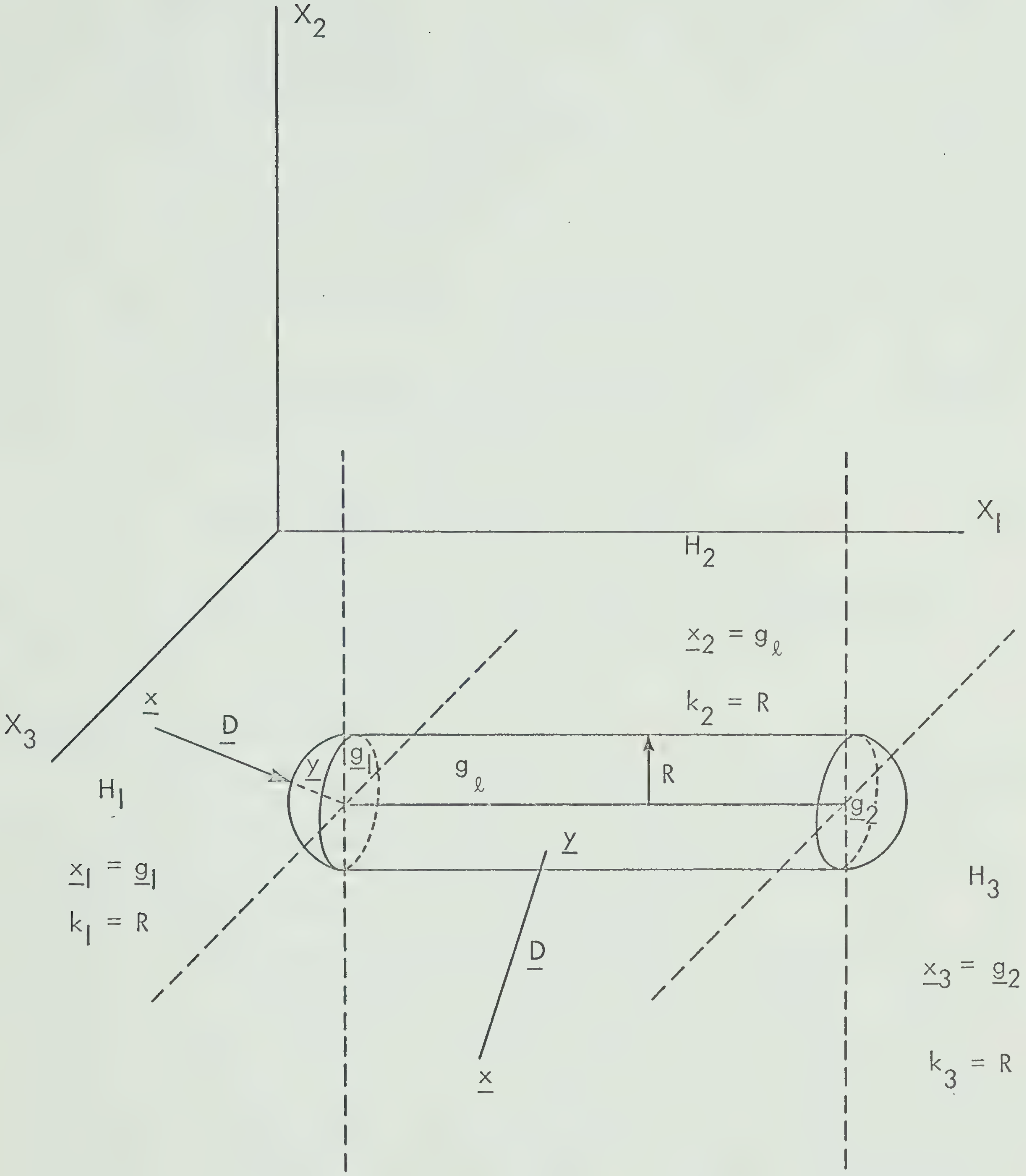


FIGURE 2.6 EXAMPLE 3: HYPERCYLINDRICAL TARGET SET WITH HYPERHEMISPHERICAL ENDS

$$\underline{a}(\underline{x}) = (x_1, c_2, c_3, \ldots, c_n) \tag{2.30}$$

$$H_{\beta 1} = \{\underline{x} | x_1 \leq \alpha_1, \underline{x} \in G^c\}$$

$$H_{\beta 2} = \{\underline{x} | \alpha_1 \leq x_1 \leq \alpha_2, \underline{x} \in G^c\} \tag{2.31}$$

$$H_{\beta 3} = \{\underline{x} | \alpha_2 \leq x_1, \underline{x} \in G^c\}$$

$$\underline{D} = \begin{cases} \dfrac{(\underline{b}_1 - \underline{x})}{||\underline{b}_1 - \underline{x}||} (||\underline{b}_1 - \underline{x}|| - R) & \underline{x} \varepsilon H_{\beta 1} \\ \dfrac{(\underline{a}(\underline{x}) - \underline{x})}{||\underline{a}(\underline{x}) - \underline{x}||} (||\underline{a}(\underline{x}) - \underline{x}|| - R) & \underline{x} \varepsilon H_{\beta 2} \\ \dfrac{(\underline{b}_2 - \underline{x})}{||\underline{b}_2 - \underline{x}||} \ ||\underline{b}_2 - \underline{x}|| - R) & \underline{x} \varepsilon H_{\beta 3} \end{cases} \tag{2.32}$$

In this case there are three regions of state space with different attracting functions.

The Terminal Condition

The purpose of the preceding discussion has been to develop arguments for the terminal condition, the requirement which ensures that the cost functional J' of (2.7) is minimized. It is required that $\underline{D}(C, \underline{n})$ be brought as far as possible in the direction $\underline{D}^+$. One of the necessary conditions is the transversality condition, that the costate vector be normal to the tangent plane to the target set. The consideration of an infinitesimal hypersphere around a point at which the tangent plane does not exist extends the transversality condition to such points, within tolerances imposed by computational considerations.

Suppose $\underline{D}^*$ is chosen to make J' minimum. Then, for all possible $\underline{D}$, $k(\underline{D}^*, \underline{D}^*) \leq k(\underline{D}^*, \underline{D})$ (2.33)

or $\quad ||\underline{D}^*|| \le ||\underline{D}||$. (2.34)

Thus $\underline{D}^*$ is the vector to the target set resulting from the terminal state $\underline{x}_T{}^*(C)$ closest to the target set.

The requirement for finding $\underline{D}^*$ is that J' be minimized. From the transversality condition a necessary condition for this is

$$\underline{p}_T(C, \underline{\eta}) = k\,\underline{D}(C, \underline{\eta}) \quad (2.35)$$

since $\underline{D}(C, \underline{\eta})$ is normal to the tangent plane to the target set or to the infinitesimal hypersphere around appropriate points of the target set. (2.35) is called the terminal condition in this thesis.

Application of the Terminal Condition

The extremal control generated by the terminal condition is found in the following manner. At cost C = 0, the value for $\underline{\eta} = k\underline{D}$ is assigned, which satisfies (2.35) since C = 0 implies t = 0. A new cost is formed by incrementing the cost by ΔC. Beginning with the value of $\underline{\eta}$ from the previous step, a new value is found which satisfies (2.35). This process is repeated until the terminal state is close enough to the target set. In this manner, the terminal state at each cost C has been brought as close as possible to the target set. For the case where the reachable set is convex and for the case where the local minimum of J' which first appears eventually leads to the optimal solution, this strategy will find the optimal solution. These considerations will be discussed later in this chapter in section 2.4.

The search strategy by which the correct value of $\underline{\eta}$ is found at each cost C consists of a steepest descent search applied to the minimization of a terminal error norm.

The error norm used is of the following form. The terminal condition required that (2.35) is satisfied. This will be so if and

only if

$$\frac{\underline{p}_T(C, \underline{\eta})}{||\underline{p}_T(C, \underline{\eta})||} = \frac{\underline{D}(C, \underline{\eta})}{||\underline{D}(C, \underline{\eta})||} . \tag{2.36}$$

Thus, the vector $\underline{E}(C, \underline{\eta})$ defined by

$$\underline{E}(C, \underline{\eta}) = \frac{\underline{p}_T(C, \underline{\eta})}{||\underline{p}_T(C, \underline{\eta})||} - \frac{\underline{D}(C, \underline{\eta})}{||\underline{D}(C, \underline{\eta})||} \tag{2.37}$$

represents the error in direction of $\underline{p}_T$. The norm of $\underline{E}(C, \underline{\eta})$ is used as the terminal error norm.

The search strategy consists of five steps. Two tests are made at various steps. Each time the system is operated, $||\underline{D}||$ is tested. If it is less than the error tolerance the search for $\underline{\eta}$ is ended. Each time $||\underline{E}||$ is calculated, it is tested. If it is less than the error tolerance, the terminal condition is considered met and the cost C is increased by the increment ΔC. The search goes back to step 1.

1. The previous successful value of $\underline{\eta}$ is used as a starting value and the system is operated. For the first step, $\underline{\eta} = k\underline{D}$ is used.
2. The system is operated n times with steps $\Delta\eta$ in each of the (Cartesian) coordinates of $\underline{\eta}$.
3. From these n + 1 values of the terminal error norm, the negative of the gradient of the terminal error norm, $-\nabla||\underline{E}||$, is found, as a first order approximation. This is the direction of steepest descent of $||\underline{E}||$.
4. The value $\underline{\eta}$ is modified by the addition of the term $(-\nabla||\underline{E}||)\Delta\eta$ and the system is operated.
5. Step 4 is repeated until $||\underline{E}||$ no longer improves. When this occurs, step 4 is reversed by subtraction of the correction term

to give the last value of $\underline{\eta}$ which improved $||\underline{E}||$. Then the search returns to step 1.

The flow chart in Figure 2.7 shows the search logic.

2.3 Time-Varying Target Sets

The natural extension of the preceding section is to a time-varying target set, using the terminal time T each time the system is operated as the time for defining the target set in finding a value for $\underline{D}$. That is, if the target set is a time-varying set, G(t), then when the system has been operated

$$\underline{D}(C, \underline{\eta}) = G(T) - \underline{x}(C, \underline{\eta}) \qquad (2.38)$$

Used in this manner, the correct final time is an additional parameter dependent on determination of the extremal solution. The same arguments hold as in the previous section. In particular, at the closest approach of R(C) to the target set, there would exist a $\underline{D}(C, \underline{\eta})$ which minimizes J' from (2.7). The application of (2.35) would find the value of $\underline{D}$ for each C.

A second approach is possible. The final time is predicted and called T_f and $G(T_f)$ is used as the target set. Then T_f is modified in the iterations along with C or $\underline{\eta}$. $\underline{D}$ could be treated as a function of T_f, so that a new value of T_f would be found using $||\underline{D}|| = 0$ and an interpolation scheme. Substantial programming effort would probably be necessary to ensure an orderly automatic solution attempt.

2.4 Convergence

There are two important considerations in the convergence of the technique presented. The conditions affecting the convergence to an extremal solution are discussed, as well as the conditions under which the solution found is optimal.

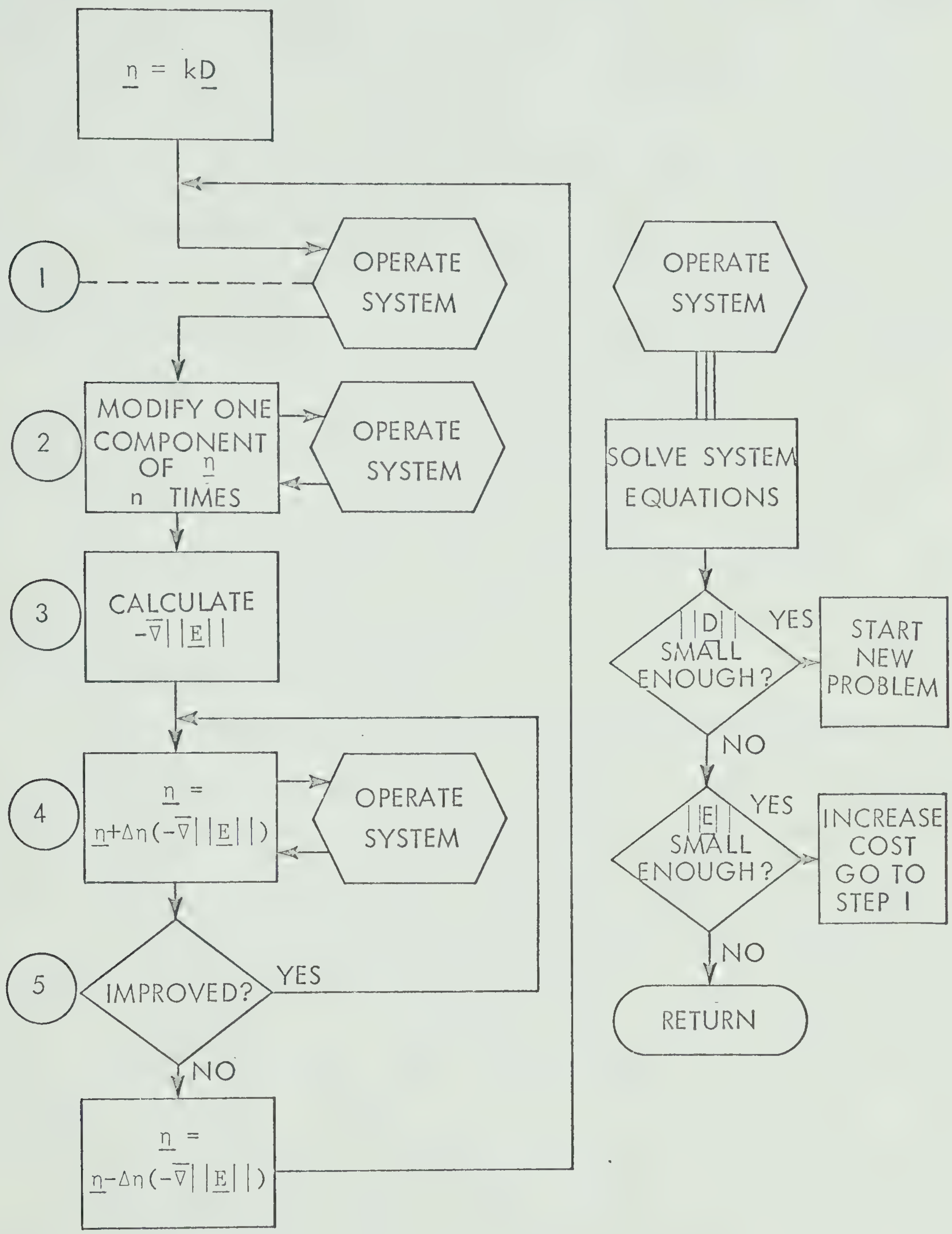


FIGURE 2.7 THE SEARCH LOGIC

It is not possible to exactly specify the requirements which would guarantee convergence to an extremal solution. As in all discrete-step methods of solving continuous problems, step size of the various incremented quantities can be of fundamental importance. The sensitivity of the final state to the initial costate often becomes great, requiring larger error tolerances in terminal matching and final state conditions. Qualitatively, it is possible to say that this method guarantees convergence in the limit as step sizes become small enough and if the costates are continuous functions of time. The latter condition, along with the continuity of the cost guarantees a continuous variation of initial costate with respect to cost. This ensures that the new initial costate will be close to the old one, for small enough changes in the cost. This heuristic argument is intended only as an explanation for the convergence of the method which has been observed.

In order to facilitate convergence the following procedure is used. Initially, rather large incremental steps are used for the search. When the terminal state is brought to within the initial error specified for the target set, the size of the steps used is reduced. The solution is then continued from that point with smaller steps, allowing close approach of the terminal state to the target set. This approach, often called "getting into the ball park", has been found to give rapid convergence in real time as applied.

This rapid convergence feature gives this method an advantage over pure analog and pure digital solutions. Pure analog solutions cannot be automatically implemented. Pure digital methods, following the strategy of a parameter search on the missing initial conditions as introduced previously, often require initial trial values close to the final solution in order to guarantee convergence without an

inordinate number of iterations, which are costly in real time and capacity. The method proposed here uses the ease of solving differential equations to advantage in order to ensure convergence to an extremal solution.

The conditions under which the extremal solution found by this method is the optimal solution have been investigated. Consider a convex target set. When the reachable set R(C) is convex, convergence is guaranteed to the optimal solution. This condition is not usually satisfied. When the reachable set is not convex, there may appear, for costs greater than some cost C_f, more than one point in R(C) $C > C_f$ which are local minima of J'. That is, there may be multiple points for which J' is locally minimum. The method proposed here will find one of them, usually the one for which $\left.\frac{d\|D\|}{dC}\right|_{C_f}$ is most negative. This method searches for the extremal solution which most rapidly approaches the target set at the cost where two or more local minima of J' come into existence in R(C). This point will be further clarified by means of an example.

Example

Consider the system given by the following specifications.

$$\underline{\dot{x}} = \begin{bmatrix} 1 \\ u \end{bmatrix} \qquad \underline{x}^{(0)} = \begin{bmatrix} -10 \\ 0 \end{bmatrix} \qquad \underline{x}^{(f)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.39}$$

$$u \in [-1, 1] \tag{2.40}$$

$$J = \int_0^{t_f} C(x_2)\, dt \tag{2.41}$$

$$C(x_2) = \begin{cases} .1 - (x_2 + 1.5) & x_2 \leq -1.5 \\ 10 & -1.5 < x_2 < 1.0 \\ 5 + (x_2 - 1) & 1.0 \leq x_2 \end{cases} \tag{2.42}$$

Examination of the system shows that the optimal solution strategy consists of the control sequence {-1; 0; 1} switching at times {0; 1.5; 8.5} and with cost 30.7. A second extremal trajectory is caused by the control sequence {1; 0; -1} switching at times {0; 1; 9} with cost 60. All trajectories remaining in the region $-1.5 < x_2 < 1.0$ are extremal with cost 100. The method presented here would, under near-perfect convergence, maintain the initial value of $\underline{\eta}$ chosen until C=100, since the initial value of $\underline{\eta}$ satisfies the terminal condition for all costs from 0 to 100. If the program, by drift or noise, should find the branch of R(C) at $x_2 = 1.0$ after cost C=10.0, it would attempt to continue with that branch until C=60. This would then be indicated as the solution. A rather advantageous set of errors in the program would be required to find the optimal branch of the reachable set. See Figure 2.8.

This example shows that the method may give quite misleading results. Further work would be desirable in clarifying the conditions causing failure of the method to find the optimal solution.

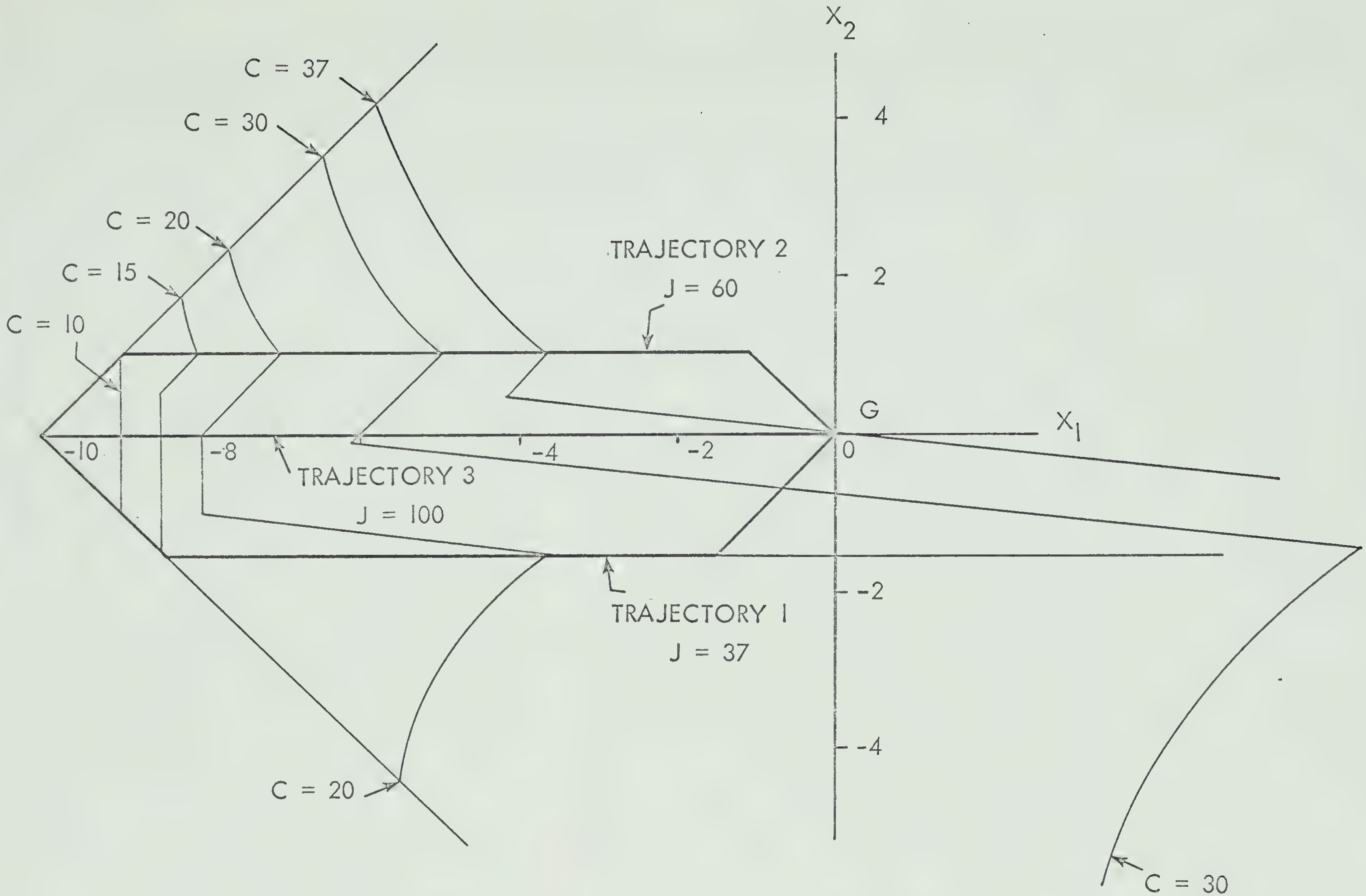


FIGURE 2.8 REACHABLE SETS FOR EXAMPLE
(RIGHT MOST BOUNDARIES SHOWN)

CHAPTER 3

METHOD OF SOLUTION

3.1 Introduction

In this chapter, the technique developed in Chapter 2 is applied to second order optimal control problems. The logic and organization of the program is described. Performance, improvements and extensions of the program are examined. In connection with this chapter, the last three appendices contain supplementary information. The techniques of hybrid computation as applied to this thesis are discussed in Appendix 2. Appendix 3 describes the hybrid computer used, an EAI TR-48 with a DEC PDP-8, in the Electrical Engineering Department at the University of Alberta. The program flow charts and the program listing assembled by PAL-D form the fourth appendix.

3.2 Organization of the Program

Implementation of the Solution Technique

Use of the technique developed in solution of the two-point boundary-value problem involves a basic change in search strategy from the direct parameter search. In the parameter search for the satisfactory unknown initial conditions, the extremal control strategy is implemented while reducing an error norm, the distance from the terminal state to the target set. The problem can be highly complicated by the lack of a final time, which causes complications in the detection of run-end conditions. In the method presented here, the magnitude of the search problem is reduced, and run-end conditions are very simple to handle. The minimization of terminal error at the projected final

time is replaced by the minimization of error in satisfying the terminal direction conditions on the terminal state and costate. At the conclusion of the search, an extremal solution to the optimal control problem is found.

The terminal direction condition, (2.35), is equivalent to the following set of conditions in the second order case.

$$\frac{p_1}{\|p_1\|} = - \frac{D_1}{\|D_1\|} \qquad \text{or} \quad p_1' = - D_1' \tag{3.1}$$

and

$$\frac{p_2}{\|p_2\|} \qquad \frac{D_2}{\|D_2\|} \qquad \text{or} \quad p_2' = - D_2' \tag{3.2}$$

where p_i' and D_i' indicate the normalized quantities as shown. The terminal error norm used is that of (2.37)

$$ND = ((p_1' + D_1')^2 + (p_2' + D_2')^2)^{\frac{1}{2}}. \tag{3.3}$$

The program attempts to bring ND to less than ER, the initially specified terminal error tolerance.

The search scheme devised is a type of steepest descent technique using the gradient of ND at a nominal initial costate. The nominal costate vector is successively perturbed in each component and the approximate gradient calculated from the corresponding values of ND. Then steps are made in the nominal costate vector in the negative gradient direction until no further improvement in ND occurs. The calculation of the gradient is then repeated and further steps in the initial costate are taken. The search ends when the value of ND at any step is less than the terminal error tolerance, ER. Then the cost is increased and the search for the correct initial costate is repeated.

The search for the extremal solution to the original problem is thus occurring as the cost is increased. It was established in Chapter 2 that continuously satisfying the terminal condition would bring the terminal state to the target set when the cost reached the minimum cost or some other extremal cost. When the state error norm

$$NX = (D_1^2 + D_2^2)^{\frac{1}{2}} \tag{3.4}$$

is found to be less than EX, the state error tolerance, the problem is considered to be solved and the solution is printed out.

Task Assignment in the Program[3],[4],[10]

Use of a hybrid computer allows a choice in the manner of execution of the various component tasks which make up the solution of the problem. In this section, brief explanation of the task assignment is given.

Input of the problem data is done by means of the digital computer, programmed to read the teletype keyboard. This allows convenient input to the program with a printed record made.

Using input data or iteration search logic and calculations, the digital computer sets the initial conditions for a trial solution run.

The analog computer solves the system state and costate equations. In so doing, it calculates the extremal control and the cost, halting the computer run when the cost exceeds the specified terminal cost.

The digital computer is used for storage of the terminal results of a trial solution. The results are tested and further values calculated according to the iteration scheme in the digital computer.

Output of the final solution is done on the typewriter of the teletype unit.

3.3 Description of the Digital Program

Language and Speed

The program is written in an assembler language for the PDP-8. The DEC language PAL-D forms the basis for the program, with a modification of their floating point language being used for the arithmetic.[6],[7] The low-level language is required by the necessity of rapid calculation and core manipulation. Some of the arguments for the use of low-level languages are presented in Bell and Griffin.[4] The explanation of the program will be concerned with blocks of instructions, not with individual instructions, in order to reach the important points and to maintain continuity. For the rest of this section, reference to Appendix 4 will be required, especially to the flow chart, Figure A-4.1.

The Main Program

START

The input message is printed and the initialization executed to prepare for data input for a new case. The input data set is typed by the operator as a string of numbers in floating point representation (decimal) and each number is stored. This is done by the READ loop using indirect addressing and address modification. The initial states are set on the appropriate digital to analog converter channels.

ENTRY

An analog computer run is made by the subroutine RUN. The value NX of $||\underline{D}||$ at the terminal time is calculated by subroutine SNX.

NX thus is the distance from the terminal state to the target set. If NX is less than EX, the specified state error tolerance, execution continues at OUTPT, which will be described following this section. If not, the terminal error norm, ND, is calculated by subroutine SND. If ND is less than ER, the terminal error tolerance, execution continues at INCOST. If not, control is transferred to the iteration scheme composed of instructions at SET1, SET2, SET3 and SET4.

OUTPT

To expedite solution, a two-stage search has been programmed. The first time that OUTPT is reached, (INDEX=0), the tolerances and step sizes are decreased and the problem restarted from the current value of initial costate. The second time OUTPT is entered, when the changed terminal error tolerance has been reached, the final values are printed out and control returned to START for entry of a new case from the keyboard.

INCOST

INCOST is entered when the terminal matching condition is satisfied. The cost is increased by DD and control is returned to ENTRY for a new search for a costate vector.

The Iteration Scheme -- SET1, SET2, SET3, and SET4.

For this section to be reached, OUTPT and INCOST have not been entered.

SET1 is entered in the first step of an iteration, when ENTRY was reached from START or INCOST, and NU is -1. The costate is called the reference costate, (N1, N2) and the value of terminal error

norm ND is saved as DM, corresponding to (N1, N2). NU is set at 0, N1 set to N1 + DE and control returned to ENTRY.

Next, SET2 is entered. ND - DM is saved as DN1, corresponding to (N1 + DE, N2). NU is set at 1, N1 reset to the reference value, N2 set to N2 + DE, and control returned to ENTRY.

When SET3 is entered, N2 is reset to the reference value, ND -DM is saved as DN2, corresponding to (N1, N2 + DE), and NU is set at 2. The magnitude of the gradient of the terminal error norm is calculated as $M = (DN1^2 + DN2^2)^{\frac{1}{2}}/DE$ and the steps in the negative gradient direction are calculated as (-DN1/M, -DN2/M). A new reference costate is calculated as (N1 - DN1/M, N2 - DN2/M). Control is returned to ENTRY.

The final step of the iteration scheme is a repetition step, at SET4. If the value of DN is less than that previous (DM), DN is saved as DM and the correction precedure is repeated. The negative gradient steps are added to the reference costate forming a new reference costate and control is returned to ENTRY. When the value of DN returned is not less than DM, this repetition ends. The negative gradient steps are subtracted from the reference costate to give the previous (and better) value, NU is set at -1 to force the calculation of a new gradient, and control returned to ENTRY.

Subroutines

DACON

This subroutine converts the octal number in the accumulator into a voltage on the channel given by DACHN. The values -3777_8 to 3777_8 (-2047_{10} to 2047_{10}) are converted to voltages in the range -10 volts to 10 volts.

ADCON

This subroutine converts a voltage on the channel in the accumulator into an octal number in the accumulator, using the same scale as indicated in DACON.

RUN

This subroutine operates the system once. The cost and initial costates are digital-to-analog converted and set onto the initial condition channels. The analog computer is put into operate mode, after a pause for charging of the initial condition capacitors. Analog-to-digital channel 0 is then tested until the run-end condition has been met and the computer placed into the hold mode. The values of the terminal state and costate are then converted to digital values and stored.

SNX

This subroutine calculates the Euclidean norm of the vector $\underline{D}_T$ as NX. In the case programmed here, the target set is the line segment on the X1 axis with right and left boundaries, DL and IL, respectively. $\underline{D}_T$ is calculated as

$$\underline{D}_T = G - \underline{x}_T = \begin{cases} (DL - X1, -X2) & X1 > DL \\ (0, -X2) & IL \leq X1 \leq DL \\ (IL - X1, -X2) & X1 \leq IL \end{cases} \tag{3.5}$$

The original value of X1 is saved as X1R, and X1 is used for the first component of $\underline{D}_T$ in the remainder of the subroutine and in SND.

The norm of $\underline{D}_T$ is calculated as $NX = (D_1{}^2 + D_2{}^2)^{\frac{1}{2}}$. (3.6)

SND

This subroutine calculates the Euclidean norm of the terminal

costate vector. Then the terminal error norm is calculated as the value ND. This was given in (3.3)

3.4 Extensions of the Program

Higher Order Problems

The program could be conveniently extended to higher order systems than second order ones. The input, operating and output subroutines would simply require additional steps to carry the extra values to and from the analog computer. The norm calculations would be modified only by the addition of additional squared terms. A branch of the iteration scheme would be added to correspond to SET1 and SET2, where the steps are taken in calculating the gradient, in order to calculate each extra component of the gradient. The major forseeable difficulty is in the analog program, where it has been found that special and possibly digital computer controlled approaches are necessary as the order of the system increases to handle scaling problems with the costate variables.[3],[9]

Other Target Sets

Extension to other target sets which are fixed is a simple matter of modifying the subroutine SNX to properly calculate the value of $\underline{D}_T$. This is possible for the target sets discussed in section 2.2, which have a finite number of attracting regions, with difficulty directly dependent on the complexity of the target set.

The time-varying case will have only the added complexity of predicting the final time in order to use the methods suggested in section 2.3. SNX would be modified to predict the final time, or to use the current time to calculate the $||\underline{D}||$ value.

CHAPTER 4

APPLICATIONS

In this chapter, as the title indicates, the technique developed in the earlier chapters is applied to a linear second order system and a nonlinear second order system.

4.1 The Harmonic Oscillator

The undamped harmonic oscillator, extensively studied in the literature, was chosen as an example of a linear system. This system is described by the differential equation

$$\frac{d^2x(t)}{dt^2} + \omega^2 x(t) = K\,u(t),\; K \geq 0,\; |u(t)| \leq 1. \tag{4.1}$$

Using the set of normalized state variables

$$\begin{aligned} \dot{x}_1(t) &= \frac{\omega^2}{K}\, x(t) \\ \dot{x}_2(t) &= \frac{\omega^2}{K}\, \dot{x}(t) \end{aligned} \tag{4.2}$$

the state equation becomes

$$\underline{x}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) . \tag{4.3}$$

The problem of transition of the state from the initial state $\underline{x}(t_o)$ to a line segment on the x_1 axis in minimum time will be considered.

$$\underline{x}(t_o) = \underline{x}^{(o)} = \begin{bmatrix} x_1^{(o)} \\ x_2^{(o)} \end{bmatrix} \tag{4.4}$$

$$g = \{\underline{g} | \underline{g} = \alpha \underline{g}_1 + (1 - \alpha)\underline{g}_2,\; \alpha \epsilon [0, 1] \tag{4.5}$$

$$G(\underline{x}) = \{\underline{x} | \, ||\underline{x} - g|| \leq R\} \tag{4.6}$$

$$\underline{x}(t_f) = \underline{x}^{(f)} \in G(\underline{x}). \tag{4.7}$$

A detailed derivation of the exact solution to this problem is found in Athans and Falb[1]. The Hamiltonian

$$H(\underline{x}, \underline{u}, \underline{p}) = 1 + p_1 x_2 - p_2 x_1 + p_2 u \tag{4.8}$$

is minimized by

$$u(t) = - \operatorname{sgn} p_2(t). \tag{4.9}$$

The costate equation is given by

$$\dot{\underline{p}}(t) = - \frac{\partial H}{\partial \underline{x}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \underline{p}(t) \tag{4.10}$$

The following points can easily be shown to be true. The solution of the costate equation is simply that of an undamped, uncontrolled harmonic oscillator; the costate trajectories being circles about the origin. The time optimal control must be piecewise constant by (4.9). There can be an unlimited number of switchings, due to the harmonic nature of $p_2(t)$. There is no possibility of singular control.

In the solution using the origin as the target set, it can be shown that the extremal trajectories are circles about the point $(\Delta, 0)$ in state space, with $\Delta = \pm 1$ being the control specified by (4.9). The time taken to move the state along one of these circular arcs is given by the angle at the centre of the circle subtended by the arc. An analytical method of finding the time optimal solution is thus available. The angle sum gives the time, while the time to the first switching gives the initial costate.

The state and costate equations were programmed on the analog computer, with initial conditions supplied by the digital computer program. Unity scaling was used (within the arbitrary factor of $\frac{\omega^2}{K}$) so

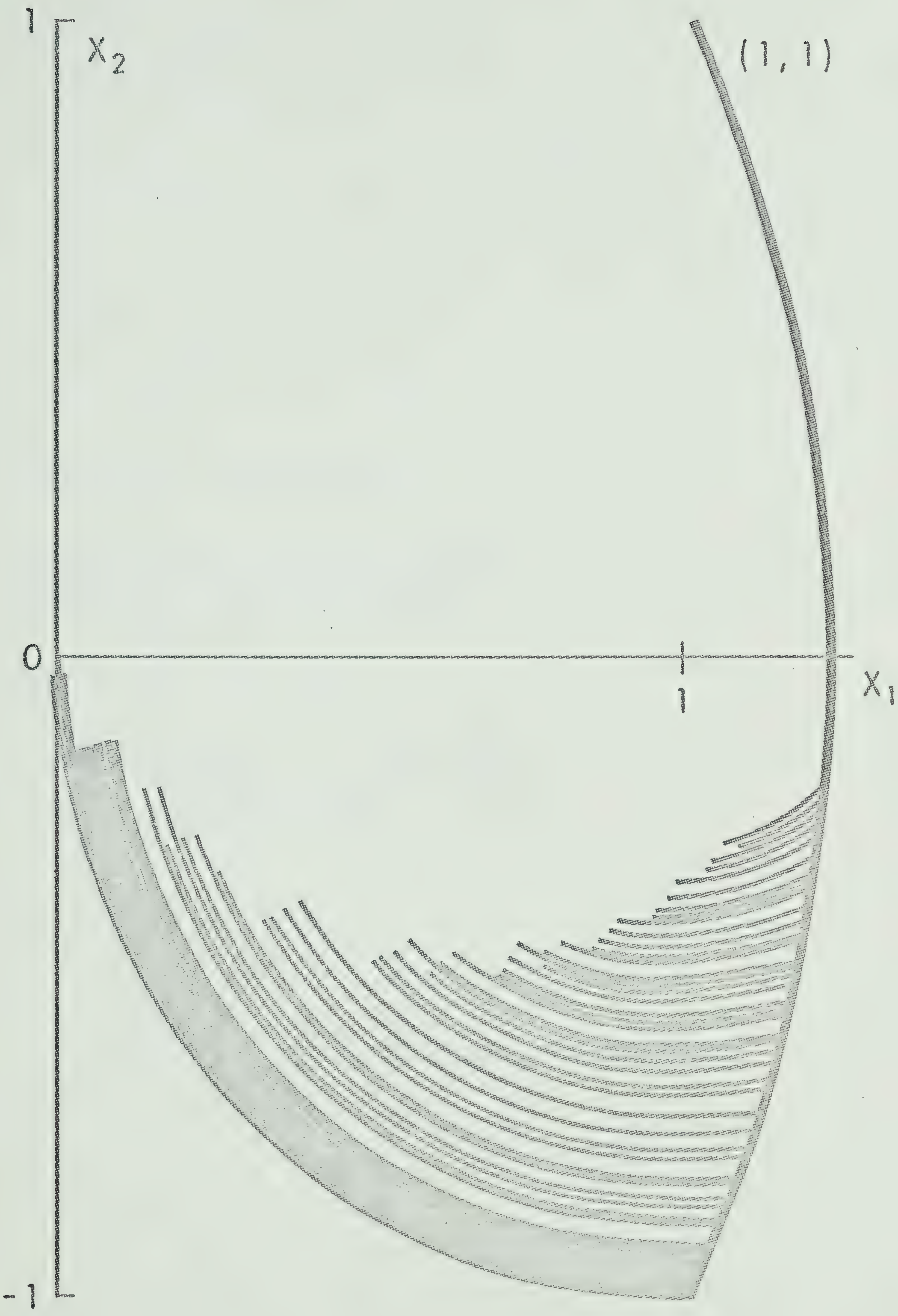


FIGURE 4.1 TYPICAL HARMONIC OSCILLATOR SOLUTION

```
HYERID COMPUTER SOLUTION


INITIAL:  COST, STATES, COSTATES
PARAMETERS:  RIGHT, LEFT TARGET POINTS, COST & COSTATE STEPS
ERROR NORMS:  STATE & TERMINAL

0.0 1.0 1.0 -3.0 -3.0 0.0 0.0 0.1 0.1 0.1 0.2


FINAL:  STATES  +    .0000  AND  -    .0097
        COSTATES  -   2.4785  AND  -   3.3939.
        COST  +   2.4999  (SECONDS).
+       122  ITERATIONS.
```

FIGURE 4.2 INPUT AND OUTPUT FOR FIGURE 4.1

that the problem was limited by the ±10 volt range of the analog computer. One volt was used as one unit for the solution.

The solution of the optimal control problem was found for a variety of cases. The results of some of these tests are given in Table 4.1. For the cases with the origin as the target set, the exact solution is included. The time taken by the hybrid computer for solution, exclusive of teletype output, is included for some of the tests. Figure 4.1 shows a typical solution of the system. It is accompanied by its input and output, in Figure 4.2. The general characteristics of these results are discussed later in this chapter.

4.2 A Bilinear System

A bilinear system considered by Moon and Mohler[11] was used as the second example. This nonlinear system is bilinear in the sense that it is linear in states and controls when separately considered. However, this property is not of any significance in the present discussion.

The problem is that of moving a searchlight and stopping at a specified angle in minimum time. The searchlight dynamics are controlled by the driving motor, which is controlled by armature current and by a brake. The losses in the system are negligible. The torque of the DC motor is proportional to the armature current.

The state equation is (from Moon and Mohler[11])

$$\dot{\underline{x}}(t) = \begin{bmatrix} x_2(t) \\ -a_2(1 + u_2(t))x_2(t) + a_1u_1(t) \end{bmatrix} \qquad (4.11)$$

where $x_1(t)$ is the angular position in radians, $x_2(t)$ is the angular velocity in radians per second, $u_1(t)$ is the armature current, $u_2(t)$ is the braking control and a_1 and a_2 are appropriate positive constants.

TABLE 4.1

Minimum Time Solutions for the Second Order Harmonic Oscillator

Part A: The Target Set is the Origin

Experimental Results

Initial State	Initial Costate Guess	Solution Time (sec.)	Number of Iterations	Final Time (sec.)	Final Costate Ratio	Final Time (sec.)	Final Costate Ratio	Costate Error %
(1,1)	(-2,-2)	7.4	117	2.48	.7568	2.50	.7499	.92
		9.0	143		.7539			.53
	(-3,-3)	5.9	89		.7514			.20
		5.7	92		.7512			.17
	(-4,-4)	6.3	101		.7500			.01
		6.1	99		.7486			-.17
		6.1	100		.7490			-.12
		6.2	101		.7498			-.01
	(-5,-5)	6.5	108		.7491			-.11
		6.9	113		.7499			0
	(-6,-6)	8.0	130		.7505			.08
		7.8	127		.7481			-.24
(1,1)	(Average)	6.83	110	2.48	.7507	2.50	.7499	.10
(4,4)	(-4,-4)	39.8	544	8.98	.9330	8.99	.9289	.44
		20.0	309		.9325			.39
		21.0	307		.9342			.57
		15.6	231		.9363			.80
	(-5,-5)	17.2	250		.9356			.72
		19.4	284		.9347			.62
		18.2	267		.9363			.80
		12.8	192		.9362			.79
	(-6,-6)	16.6	240		.9319			.32
		18.0	257		.9335			.50
		16.0	234		.9365			.82
		15.8	233		.9352			.68
(4,4)	(Average)	19.4	279	8.98	.9347	8.99	.9289	.62
(4,4)	(Average 9 other runs)		381	8.98	.9341	8.99	.9289	.56
(5.405,0)	(-5,0)		222	8.30	7.30	8.32	6.73	8.48
			374	8.30	7.01			4.16
			134	8.32	7.16			6.40
(0,6.325)	(0,-6)		1766	9.74	.01278	9.73	.00931	37.27

TABLE 4.1

Part B: The Target Set is a Line Segment on the x_1-axis

Experimental Results

Initial State	Initial Costate Guess	Target Segment	Final x_1 Value	Number of Iterations	Final Time (sec.)	Final Costate Ratio
(4,4)	(-3,-4)	[-1,1]	-.42	403	8.90	.807
		[-1,1]	-.703	553	8.90	.735
		[0,1]	0	640	8.98	.942
		[-.5,1]	-.5	365	8.90	.789
		[-.4,1]	-.4	953	8.92	.820
		[-2,1]	-.776	1179	8.90	.717

For $i = 1, 2$ $|u_i| \leq 1.$ (4.12)

The initial state is given as $\underline{x}^{(o)}$ and the final state is required to be $(0, 0)^T$ in the original problem. The Hamiltonian is

$$H(\underline{x}, \underline{u}, \underline{p}) = 1 + p_1 x_2 + p_2(a_2(1 + u_2)x_2 + a_1 u_1) \tag{4.13}$$

and the costate is given by the equation

$$\dot{\underline{p}}(t) = \begin{bmatrix} 0 \\ -p_1(t) + a_2(1 + u_2)p_2(t) \end{bmatrix} . \tag{4.14}$$

Minimization of the Hamiltonian with respect to the control yields the control functions

$$\begin{aligned} u_1(t) &= -\text{sgn } a_1 p_2(t) = -\text{sgn } p_2(t) \\ u_2(t) &= \text{sgn } a_2 p_2(t) x_2(t) = \text{sgn } p_2(t) x_2(t) \end{aligned} \tag{4.15}$$

The problem was solved using the following values:

$$t_o = 0;\ a_1 = 2;\ a_2 = 1;\ x_1^{(o)} = 5;\ x_2^{(o)} = 10. \tag{4.16}$$

The state and costate equations were programmed on the analog computer, along with the control functions. The initial conditions were supplied by the digital computer program. Unity scaling was used, thus $x_2^{(o)}$ was set as close to full scale as possible, 9.997 volts, one bit short of 10 volts.

The case described was studied in depth, since the problem was almost full scale, and since comparison with the results of Moon and Mohler was possible. Cases using target sets other than the origin did not add any additional information to the study.

The results of some of the tests are recorded in Table 4.2. A typical solution is shown in Figure 4.3, with its input and output in Figure 4.4. The general characteristics of these results are discussed in the following section.

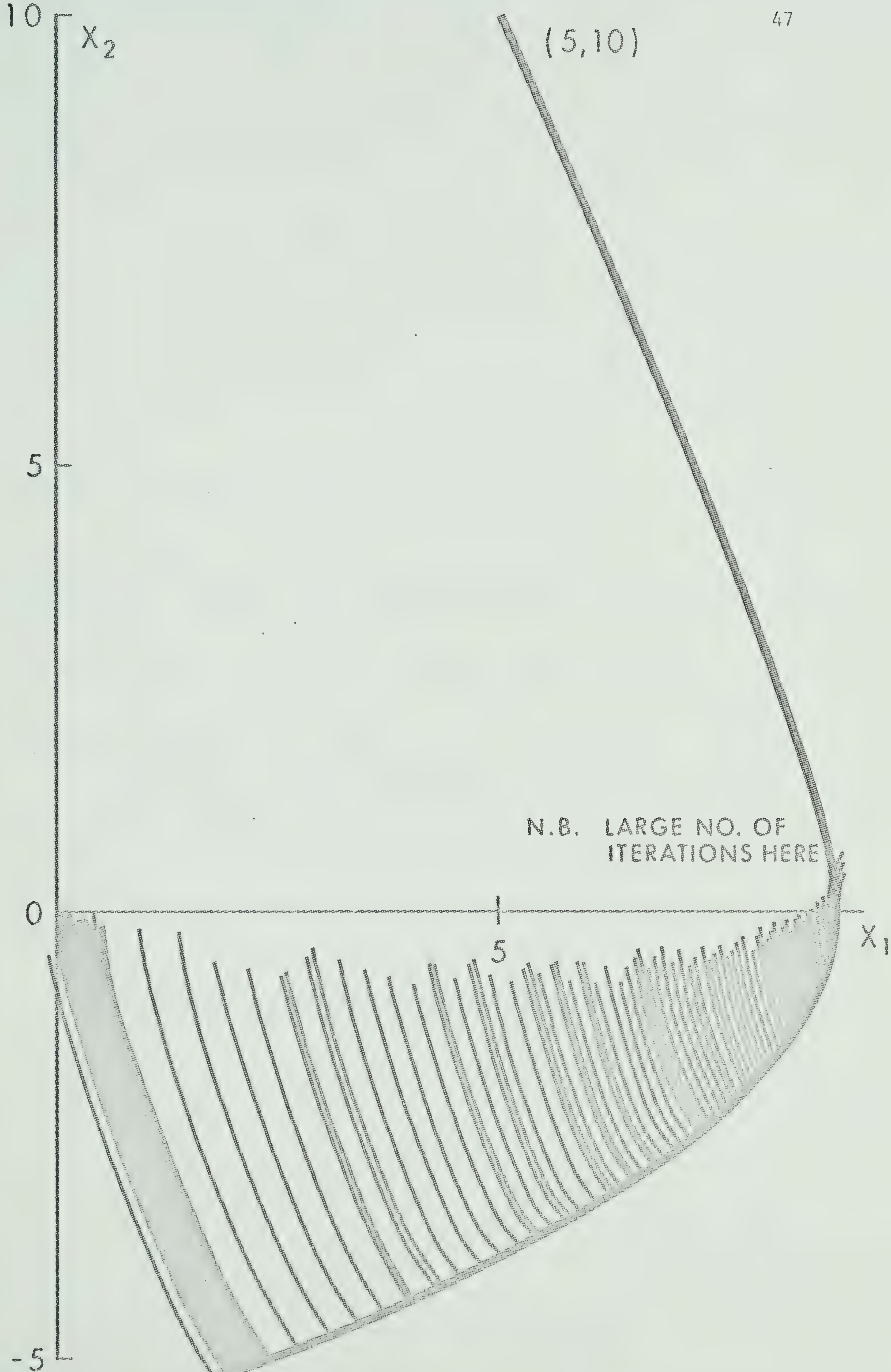


FIGURE 4.3 TYPICAL BILINEAR SYSTEM SOLUTION

```
HYBRID COMPUTER SOLUTION


INITIAL:  COST, STATES, COSTATES
PARAMETERS:  RIGHT, LEFT TARGET POINTS, COST & COSTATE STEPS
ERROR NORMS:  STATE & TERMINAL

0.0 5.0 9.997 -3.0 -6.0 0.0 0.0 0.1 .025 .2 .15


FINAL:  STATES  +    .0146  AND  +    .0097
        COSTATES  -   1.3401  AND  -   6.3496.
        COST  +   4.9399  (SECONDS).
+      615  ITERATIONS.
```

FIGURE 4.4 INPUT AND OUTPUT FOR FIGURE 4.3

TABLE 4.2

Minimum Time solutions for a Second Order Bilinear System

Experimental Results

Initial State	Initial Costate Guess	Target Segment	Number of Iterations	Final Time (sec.)	Final Costate Ratio
(5,10)	(-3,-6)	[0,0]	5077	4.90	.2132
			3235		.2123
			3518		.2132
			612		.2151
			713		.2132
			1146		.2128
			680		.2141
			514		.2141
			637		.2143
			1640		.2143
			4144		.2165
			1193		.2149
			2040		.2165
			1099		.2170
			1870		.2208
			434		.2228
			892		.2199
			2875		.2199
			810		.2199
			2889		.2248
			788		.2284
			738		.2304
			868		.2359
			984		.2357
			1216		.2361
			1061		.2558
			1104		.2456

4.3 Discussion of Results

The main objective of this research has been more to examine the effectiveness and validity of the technique developed rather than to compile solutions for a large number of examples. Consequently, only a few of the large number of cases treated were studied in depth. The general conclusions, however, are based upon the results of these detailed studies and upon the information gained by trying the other cases.

It is well known that the costate equation must be linear in the costate, so that the costate can be found only to within a scale factor with its sign determined. For this reason, the ratio of costates is reported here. In order to consider values in the full dynamic range of the costate to ensure that the more influential component was used half the time, both the costates were varied. This also tested iteration techniques for handling two simultaneous searches for future applications.

The tests on the harmonic oscillator showed that a rapid solution was indeed possible using the program developed to implement the technique. The solution times were in the order of ten seconds. The study of this system indicated the severity of step size effects. The use of large steps was found to result in fewer iterations until the target set was approached closely. Then the method would usually fail to converge, since the terminal states had become highly sensitive to initial costate changes and large steps were being used. On the other hand, small step sizes usually guaranteed convergence, but usually with a large number of iterations. These effects suggested the implementation of two stages of search, the first one with large steps until

the target set was approached closely, and the second one with smaller steps until the solution had converged sufficiently. This strategy was adopted, using the arbitrary factors of five for the cost and costate steps, and terminal condition error tolerance, and ten for the terminal error. In general, it was evident that step size choice was a compromise between speed of solution and risk of failure to converge.

The cases tested showed a lack of close repeatability. There were three factors in this. The solution accuracy in "fast-time" operation is probably closer to 1% than to .1%. This meant that the terminal values fluctuated from run to run. The discrete steps of the analog data after conversion to digital data was in effect quantization into steps which grew in proportion as the value became smaller. Consider a quantity which approaches zero. The number of bits representing it decreases, until each bit is a large part of the quantity. An analog voltage less than .01 volts in magnitude can be only one of five values: -2, -1, 0, 1, 2 (octal). The third factor was the affect of rounding the computed costate changes for digital to analog conversion. This caused steps in the initial costates rather than continuous changes.

One factor which appeared to enter into the cases studied in depth was the switching error. The $(1, 1)^T$ case had only one switching, whereas three switchings occured in the $(4, 4)^T$ case. The finite speed of the comparator used for switching (7μs) probably contributed to the greater mean error in the costate ratio for the latter case.

The bilinear system showed a major difficulty in application to some problems. The progress of the search was arrested for a large number of iterations while an abrupt change in the initial costate was

made. This brought into question the assumption of Miura et al[10], namely that the initial costate changes continuously with increasing cost. It became obvious that this assumption failed whenever the mode of solution being followed eventually led away from the target set, at which time a new mode for the control strategy had to be found. The program was able to find the new mode but the technique was not designed for jumps in the initial costate and thus did not handle them efficiently.

A wide range of solution times (not tabulated) was observed due to the jump in the initial costate, from 25 to 100 seconds. In the cases examined, a large number of iterations occured, occupying a substantial portion of the solution time, while the initial costate was changed to allow continuation. This is noted in Figure 4.3.

Although difficult to compare directly, Moon and Mohler's results were available for comparison. They reported that the system was solved using their method with an IBM 360/75J at the University of California at Los Angeles in approximately 4.05 seconds per run, using less than 64K bytes (with 32 bits per byte).[11] The PDP-8 used just over 1K bytes, with a little help from the TR-48 analog computer. It is probably safe to consider the cost of using the 360/75J as somewhat more than ten times that of the PDP-8-TR-48 hybrid, while the solution time was no more than ten times better.

CHAPTER 5

COMMENTS AND CONCLUSIONS

5.1 Improvements in the Program

The conclusions reached in this thesis are based somewhat on the experimental work, and are thus influenced by its imperfection. The possible improvements in the program are discussed here in order to moderate some of the adverse affects of the particular implementation of the technique.

The most important problem which depended on the program was concerned with the search strategy. With the "a posteriori" knowledge that jumps in the initial costate are possible, the program could employ a strategy which used very large steps at the appropriate time to find a new mode of extremal control. During normal searching on the terminal error condition, the steps used could be variable depending on the current success or failure of the gradient search, according to the planned strategy. It would be desirable to increase the step size whenever the convergence appears satisfactory and to decrease the step size if the steepest descent iterations are not succeeding and at the final stages of meeting the terminal error and target conditions.

The total execution time could be reduced by about twenty-five per cent by the use of machine language programming exclusively in the solution iterations. Since all values communicated can be only one word long, no significant use of the Floating Point System was made in the calculations. One word arithmetic, possibly with provision for

some round-off controls could just as well have been used. The programming effort would be increased, however.

5.2 Summary

The technique initiated by Miura, Tsuda and Iwata[10] for the solution of optimal control problems on the hybrid computer has been examined and extended in this thesis. The algorithm for more general target sets was derived and shown to work on a second order system.

It was shown that the assumption, made in Miura et al[10], that the extremal solution which is initially closest to the target set is the optimal one, does not always hold. A counter-example was described in Chapter 2 to support this statement. In addition, it was shown experimentally that the assumption that changes in initial costate are continuous with increasing cost is invalid, and it was to be expected that shifts or jumps in the initial costate would occur as cost increased. This was caused by the need for new modes of control which were not initially minimal-distance ones.

The hybrid computer program was shown to be usable on some examples, even when the unexpected shift in the initial costate appeared. The efficiency of the program could be improved for regular use of the program. The use of "fast-time" integration was necessary to make solution times reasonable. The comparison between the solution from Moon and Mohler[11] and that in this thesis is difficult to make, but the product of capital cost and solution time for the hybrid computer is probably smaller than for the IBM 360/75J. Satisfactory solution of these systems on a PDP-8 alone would be slow or impossible because of the small memory available.

5.3 Suggestions for Future Work

The technique developed here could be used in the solution of optimal control of deterministic systems modelled on the analog computer. The validity of the algorithm for a particular system would have to be established because the algorithm is not always applicable. The convergence to an extremal solution could be guaranteed by improvements in the program to handle shifts in the initial costate.

The limitations of this technique seem to indicate that more work should be done in order to establish sufficient conditions for the solution of given classes of problems. Using the method on more complex systems without establishing its validity might prove futile. The application to time-varying target sets should be made. In the case of systems for which the assumptions made in this thesis are valid, the application of some other techniques, which do not use the costate and the minimum principle, such as perturbation techniques, should be examined.

Since it has been found that the technique works for the second order harmonic oscillator, the relationship between the ideal (undamped) case and non-ideal cases (e.g., positive and negative damping) could be examined.

The on-line control of a simple real system might be attempted by simulating the system on the hybrid computer, using the model to determine the optimal control strategy and then using the hybrid computer to apply the strategy to the real system.

It is proposed to submit the significant results in this thesis for publication in the IEEE Transactions on Computers.

BIBLIOGRAPHY

1. Athans, Michael, and Peter L. Falb: Optimal Control; McGraw-Hill Book Company; New York, 1966.
2. Balakrishnan, A.U., and L.W. Neustadt(Editors): Computing Methods in Optimization Problems; Academic Press; New York, 1964.
3. Bekey, George A., and Walter J. Karplus: Hybrid Computation; John Wiley & Sons, Inc.; New York, 1968.
4. Bell, D., and A.W.J. Griffin (Editors): Modern Control Theory and Computing; McGraw-Hill Publishing Company Ltd.; London, 1969.
5. Bryson, A.E., and W.F. Denham: "A Steepest-Ascent Method for Solving Optimum Programming Problems"; Journal of Applied Mechanics, Transactions of the ASME; Vol. 29, pp. 247-257, 1962.
6. Digital Equipment Corporation: "Floating Point System Programming Manual, 8-5-S"; Maynard, Mass., 1965.
7. Digital Equipment Corporation: "PAL-D Disk Assembler, DEC-D8-ASAA-D"; Maynard, Mass., 1968.
8. Lee, E. Bruce, and Lawrence Markus: Foundations of Optimal Control Theory; John Wiley & Sons, Inc., New York, 1967.
9. McLeod, John (Editor): Simulation; McGraw-Hill Book Company; New York, 1968.
10. Miura, Takeo, Junji Tsuda, and Junzo Iwata: "Hybrid Computer Solution of Optimal Control Problems by the Maximum Principle"; IEEE Transactions on Electronic Computers; October, 1967, pp. 666-670.

11. Moon, Sang Fi, and Ronald R. Mohler: "Optimal Control of Bilinear Systems and Systems Linear in Control, EE-164(69) NSF-118"; Bureau of Engineering Research, The University of New Mexico, Albuquerque, New Mexico, 1969.

12. Porter, William A.: Modern Foundations of Systems Engineering; The Macmillan Company; New York, 1966.

13. Rozonoer, L.I.: "L.S. Pontryagin Maximum Principle in the Theory of Optimum Systems. I" Automation and Remote Control (USSR); Vol. 20, 1959; pp. 1288-1302.

APPENDIX 1

Some Definitions and Results in Optimal Control Theory[1]

1. A Dynamical System

A dynamical system S is the composite concept consisting of sets T, Σ, Ω, and U, a variable $\underline{x}(t)$, and a function $\underline{g}$ such that the axioms listed are satisfied. T, a subset of the real numbers, is called the domain of the system. Σ is the (metric) state space of the system. U, the input space of the system is a set of piecewise continuous functions on T with values in the (metric) space Ω. $\underline{x}(t)$ is the state variable and is defined on T with values in Σ. $(t_o,t]$ is a half-open interval in T and used as an argument denotes a segment of a function on T restricted to $(t_o,t]$. $\underline{g}$ is a function from $\Sigma \times \Omega \times T$ into E^P. $\underline{u}(t_o,t]$ is a segment of $\underline{u}(t)$ in U, called the input over the observation interval to the system. $\underline{y}(t_o,t] = \underline{g}[\underline{x}(t_o),\underline{u}(t_o,t]]$ is a corresponding output of the system, with $(\underline{u}(t_o,t], \underline{y}(t_o,t])$ forming an input-output pair of the system. $\underline{x}(t) = \underline{\phi}[t;\underline{u}(t_o,t], \underline{x}(t_o)]$ describes the trajectory of the system on $(t_o,t]$, starting from $\underline{x}(t_o)$ and generated by input $\underline{u}(t_o,t]$.

It is assumed that the state at t_o, $\underline{x}(t_o)$, and the control $\underline{u}(t_o,t]$ specify a uniquely defined output. The system is nonanticipative since future values of the input do not affect $\underline{y}(t_o,t]$.

It is assumed that every input-output pair has a corresponding state history $\underline{x}(t_o,t]$ in Σ.

It is assumed that $\underline{g}$ and $\underline{\phi}$ are smooth in their arguments so that small changes in the input cause small changes in the output and

trajectory of the system.

It is assumed that the transition function $\underline{\phi}$ satisfies the transition, or semigroup, condition in that the initial condition corresponds to the starting point of the trajectory; and if the input takes the system from $\underline{x}_o$ to $\underline{x}$ along a trajectory, and if $\hat{\underline{x}}$ is a state on the trajectory, then the input will take the state from $\hat{\underline{x}}$ to $\underline{x}$.

2. Finite Dimensional

A dynamical system S is finite dimensional if the state space is a Euclidean space, $\Sigma = E^n$; and Ω is a Euclidean space, $\Omega = E^m$, with $m \leq n$. n is the dimension of the system.

3. Continuous-Time

A dynamical system S is continuous-time if T is an open interval of the real numbers.

4. Differential

A dynamical system S is a differential system if the state is given by a solution of a differential equation system $\dot{\underline{x}}(t) = \underline{f}(\underline{x}, \underline{u}, t)$ $\underline{x}(t_o)$ as initial condition and $\underline{g}$ is a continuous function of its arguments.

APPENDIX 2

Some Aspects of Hybrid Computation

Hybrid computation is discussed from the viewpoint of optimal automatic control studies with the plant simulated in the analog computer. This restricts the scope of this appendix, which is not intended as a tutorial account for the uninitiated, to the area of concern of this thesis. The object is to summarize some of the aspects of hybrid computation examined by the author and to indicate some generalizations which have been made.

Bekey and Karplus define a spectrum of hybrid computing techniques and systems, ranging from pure analog computers with digital logic to pure digital computers with analog inputs. They state, "The most extensive and powerful of existing hybrid computer systems are comprised of general-purpose digital computers and general-purpose analog computers."[3] Bell and Griffin concur, terming this type of system "the most flexible hybrid computing facility".[4]

Specifically, such a hybrid computer consists of the analog and digital computers mentioned, with a communication link between them. The function of the communication link is to transfer analog, digital and logical signals between the computers, making the appropriate conversions. It is this type of hybrid computer which is implied in this appendix.

A-2.1 Inter-Computer Communication

In a hybrid computer system the communication link often becomes the "weak link" in the chain. The number of channels of analog-to-digital and digital-to-analog conversion (hereinafter termed ADC and

DAC, respectively) which are available determine the amount of information which can be transferred between computers. The speed of multiplexing determines the rate of transfer (with fast languages). The availability of extra logic signal channels can simplify many facets of computation without tying up an information channel. Mode control of the analog computer falls into this category.

The time taken by the communication link to perform the operations required may introduce a variable and significant time lag into some computations. This time lag is directly proportional to the slowness of the digital computer language used. For instance, the applications in which the digital computer is used for calculations of nonlinearities for the analog computer solution are seriously affected, especially in "fast-time" operation of the analog computer.

The effect of the communication link in speed of operation can be lessened by avoiding its use in "critically timed" operations. An analog comparator switching in 7 μs is obviously better than an ADC, a test of the value, and a DAC consuming about 70 μs, when using the switching within the analog computation. In the same category of operation is the use of an incremental conversion ADC fed directly into the accumulator of the digital computer and tested as often as possible. This technique would require about 7 μs in the PDP-8-TR-48 system, but possibly less in a digital computer with half the cycle time of 1.5 μs of the PDP-8. In any event, it is clear that using the digital computer in ongoing analog calculation requires study of the effects of computation times, both fixed and variable. There are several applications, however, in which the digital computer results

are used only when the analog computer is in hold mode. This is the technique used in this thesis.

Communication is also affected by the discretization of continuous analog data by the ADC. This is further discussed in the following section of this appendix.

A-2.2 Accuracy

Accuracy requirements in hybrid computation are determined by a somewhat different set of conditions than those affecting digital computation. In particular, the solution of a set of differential equations programmed on an analog computer will converge to a real solution (of the system programmed); whereas one has no guarantee that a given method and step size will converge to a solution in a digital computer solution of the same set. The precision used in digital computation is made necessary to some extent by the methods chosen for rapid solution, at the price of guarantees of convergence.

In simulations of many physical systems, accuracy in the order of one per cent is completely acceptable. The non-exact nature of the differential equations, parameters, input signals and outputs precludes the use of six to sixteen significant figures which are often necessary in the solution of the system on a digital computer. The limit of accuracy for analog computers is .01%, but the high speed transistor amplifier models are limited to .1%. In high speed operation this may become 1% as switching times and capacitive effects become more significant.

Since the costate equations are unstable for stable state equations systems, the magnitude of the initial condition of the costate must in general be sufficiently small to prevent the costate

system from reaching the limits of the analog computer amplifiers; yet it must be as large as possible to cope with discretization problems.

The discretization of the communicated data has an important effect here. A 12 bit word implies a step size of .049% of the analog computer full scale (10 volts for most transistor systems). For a costate value which increases with a dynamic range of 100 over the solution time, this would make each step 5% of full scale, a significant discretization error.

The analog computer, DAC and ADC must be calibrated properly to ensure maximum accuracy. The nature of the applications considered in this thesis are such that drift of the analog component characteristics does not have a significant effect, since run times are very short, and initial conditions are reset from the DAC for each run. Along with calibration, loading effects must be considered since fast, large bandwidth amplifiers are used. The comparator output may be susceptible to loading error, for example.

The accuracy of calculations in the digital computer must be considered, but in applications such as found in this thesis, the calculation requirements are few and are more or less independent of previous runs. This precludes the buildup of round-off and numerical instability errors.

A-2.3 Speed

A fundamental consideration is speed of solution. The speed of solution of a hybrid computer method must be of the same order as a good digital computer method to be competitive. This requires the use of the fastest possible techniques in both the analog and digital

portions of the hybrid system.

Most modern analog computers are constructed with two normal speeds of integration. The "slow-time" mode uses integrators of unity gain as the basic unit. The "fast-time" mode uses a gain of 500 or 1000. Normal usage of this mode is in repetitive operation with initial condition and operate modes cycled from 10 to 1000 (or more) times per second. If the mode control is accessible, the digital computer can control the analog computer, replacing the timing unit, or individually controlling sets of integrators. For studies such as found in this thesis, the high speed mode must be used whenever possible to decrease run time.

One important consideration when the digital computer is used to control the analog runs is that enough time must be allowed for the initial condition capacitors to be charged to the specified levels. To set initial conditions to within .05% of full scale requires 7.6 time constants of the charging network. The finite switching speed of the comparators and the mode controls must also be taken into account. In many computers, the hold mode is switched by relays, which may necessitate the use of track-and-hold amplifiers to stop critical values at the instant desired, faster than the computer can be switched into hold mode.

The communication link is likely to constrain the speed of solution in applications in which conversions form a large part of the calculation. Some systems are set up with relay multiplexing, which is almost useless at high speed operation. Solid state multiplexing and low settling time are important in this type of calculation.

The digital computer speed begins to become significant when high speed analog computation is used. One will often have the choice of different levels and types of languages. These are summarized here, specifically with reference to the PDP-8. An important consideration is execution time, since the program will be repeated many times for one solution.

The slowest languages are the interpretive ones, such as FOCAL from DEC. The operating system interprets each command and each variable when encountered in execution, performs the command using any subroutines necessary, and then reads and interprets the next instruction. In FOCAL, 10 ms is the minimum amount of time required to read a command and execute it. ADC and DAC take about 40 ms. Obviously, such a language is too slow for rapid solutions.

Compiler languages are much faster in execution. A language such as FORTRAN has a compiler which changes the instructions into machine language commands, stores variables, checks for errors and produces an "object program" which is the actual set of machine language commands. These languages are usually somewhat inefficient in that the compiler is not able to use the techniques a human programmer might use, and in that every program must have allowances made for situations which may be encountered in different types of programs. In the two types just mentioned, programming effort is lowest. Instructions can be efficiently written, with the task of sorting out the machine language sequences necessary to execute them left to the compiler.

A low level language which is a compromise between the higher level languages so far mentioned and the very low level languages to be

mentioned is illustrated by DEC's Floating Point System.[6] This language contains instructions analogous to assembly language instructions, which are assembled in the same manner. They allow the programmer to program floating point operations on numbers consisting of three words, one for the power of 2 and two for the fraction between .5 and 1.0. Thus 5.0 is stored as $0003\ 2400\ 0000_8$, or $.625_{10} \times 2^3$. Some arithmetic operations and functions are available and input and output routines are used, as supplied or added by the author. The PAL-D Assembler[7] is used to assemble the symbolic instructions into machine language code, with most of the organization of the program already done by the programmer. Further discussion is contained in the third appendix.

The lowest level of programming language is represented by DEC's PAL-D Assembler in its basic form. At this level each machine language instruction is represented by a symbolic code. Variables are addressed as names, and addressing and assignment of space is usually the concern of the programmer. Some automatic assignment of addresses is possible. The programmer, through the exertion of a substantial amount of programming effort, may be able to develop a very efficient program as far as execution time is concerned. The programming effort required, once basic skills at assembly language programming are acquired, may be justified by the efficiency, but a compiler language should be available for many applications and especially for prototype programs. The Floating Point System must be considered to be an assembly language in this respect, differing from the simple assembly languages only in the availability of a number of subroutines for useful operations.

The discussion of computer languages has been oriented to the PDP-8, which was used by the author. Other small computers with 4K or 8K of core also have the same type of possibilities and most of the preceding discussion will apply.

One further point is of importance. The output from the hybrid computer should be considered as a factor in the utility of applications. High speed solutions may be displayed on a storage oscilloscope, with photographs providing the "hard" copy which may be desired. The monitoring of the solution with the oscilloscope allows operator intervention in the case of unforeseen problems. Should the hybrid computer be used in an application where numerical values are required, then the (very) slow teletype or electric typewriter output should be used. In applications such as those in this thesis, the amount of time for printing out the few results is of the same order of magnitude as the solution time. Clearly, in a situation where the hybrid computer is used to solve a control strategy problem for a real-time system, the result should be immediately applied to the system, after which the pertinent permanent record should be made, possibly on a shared time basis with the real-time control program.

APPENDIX 3

Description of the Hybrid Computer

The hybrid computer used for this work is located in the Department of Electrical Engineering at the University of Alberta. It consists of one TR-48 solid-state analog computer, an interface, and one PDP-8 solid-state digital computer. The computer is operated in an "open-shop" manner.

The TR-48 is a medium capacity general-purpose analog computer, manufactured by Electronics Associated, Incorporated, West Long Branch, N.J. Its range is from -10 to + 10 volts with chopper-stabilized transistor operational amplifiers. The computing components consist of the following: 18 integrators, 36 summers, 5 electronic bipolar multipliers, 4 electronic comparators, 8 electronic switches, 60 pots, 8 servo-set pots, 8 feedback limiters and 2 twenty-segment diode function generators.

The mode of each pair of integrators can be controlled individually. Mode control is effected by electronic switching between operate mode and initial condition mode and by relay switching between these modes and hold mode. In repetitive operation mode, the mode control logic signals are cycled between operate and initial condition, with ten milliseconds initial condition mode and from ten to five hundred milliseconds for operate mode. This particular cycle can be replaced by any other scheme desired using the digital computer for control. It is this type of operation which is referred to an "fast-time", since in repetitive operation (whether or not internally cycled) the speed of the integrators is increased by five hundred. "Fast-time" operation

allows the high speed solution of differential equations, and makes the iterative solution of ordinary differential equations practical.

The interface or communication link is based on Redcor Series 610 Linkage System.

Twelve digital bits are converted to ±10 volts of analog information by one of sixteen digital to analog converters. An analog voltage from one of twenty-four channels converted to thirteen bits (twelve being used) by the analog to digital converter by means of a solid-state multiplexer.

The remainder of the interface consists of logical and timing facilities designed and implemented in the Department of Electrical Engineering. The only one of concern here is the analog mode control facility, which translates the command from the digital computer into the appropriate logical values to bring the appropriate mode.

The PDP-8 is a small general-purpose digital computer manufactured by Digital Equipment Corporation, Maynard, Mass. It has 4K (4096) twelve bit words of core memory, operating with a cycle time of 1.5 μs, two TU-55 Dectape magnetic tape units and a 32K disc memory. It has a full complement of hardware machine language instructions and can service sixty-four peripheral devices requiring three commands each. The analog mode control commands, and the commands used in ADC and DAC are examples of these. The PDP-8 has the extended arithmetic option, which improves arithmetic calculation speed by the use of hard wired instruction. An ASR-33 teletype provides teletype input and typewriter output to the computer by means of appropriate programs.

As important as the hybrid computer hardware is, the software must be flexible in order to make use of it. The PDP-8 is supplied with

a wide range of software from assembly to compiler to interpretive languages. In this thesis the PAL-D Assembly Language[7] is used to assemble the machine language sections of the program. The reader is also referred to Introduction to Programming; Digital Equipment Corporation's general manual on programming the PDP-8 family of computers.

The Floating Point System[6] has a number of features which made it convenient to use. The program, as supplied by DEC, allows operations to be performed on three-word floating point numbers. The first word is the power of two and the other two form a twenty-four bit mantissa between one half and one in magnitude when normalized. Input and output routines are provided for the teletype. Allowance is made for the addition of subroutines which operate on the floating accumulator (FAC). The operations used in this thesis are the following:

FADD A (FSUB A) - Add (subtract) the number A to the FAC, normalize and leave in the FAC.

FMPY A, (FDIV A) - Multiply (divide) the FAC by the number A, normalize and leave in the FAC.

FGET A - Load the number A into the FAC.

FPUT A - Store the FAC into the three words starting at A.

FNOR A - Normalize the mantissa of the FAC to magnitude between .5 and 1.0.

These are supplemented by the following subroutine-type commands:

FEXT - Leave floating interpreter. Instructions following are in machine language.

SQUARE - Square the FAC.

SQROOT - Take square root of the FAC.

FIX - Convert the FAC into a one-word number at location 45.

INPUT - Read a decimal number from the teletype.

OUTPUT - Print the contents of the FAC on the teletype as a decimal number.

CRLF - Print a carriage-return and line-feed.

NEWOUT - Execute OUTPUT, then CRLF.

RNDFIX - Convert the floating accumulator contents into the nearest one word number at location 45.

Three additional subroutines are used in the program. The subroutine called by FLOAT will transform the contents of the real accumulator into a normalized floating point number. The subroutine called by MSSGE will print, on the teletype, the text stored between the first address, immediately following, and the last address, whose negative follows the first address. The message is stored two literal characters to a word by the TEXT pseudo-operator of PAL-D. The character (\) is used to execute the instruction CRLF as the message is printed. The other subroutine used is called by IN, and is the entry into the floating point interpreter. All instructions following are interpreted as floating point ones, until the command FEXT is encountered.

The speed of execution of the floating point system lies between machine language programming and interpreter language programming. The interpretation of commands is what makes the difference. The arithmetic commands (FADD, FSUB, FMPY, FDIV) take from 330 to 385 μs

to execute. In the interpreter language FOCAL, the equivalent commands take 20 to 40 ms. To set one variable equal to another (FGET A; FPUT B) takes 270 μs. In FOCAL this would take about 20 ms. In this program, an ADC takes 69 μs, and a DAC 44 μs. Using the FOCAL functions takes about 40 ms. The use of the machine language and floating point programming increases the speed of digital operations by a factor of about 500. The net effect of high-speed operation is a speed increase of 500 times. Use of only one of the techniques (i.e. using normal speed integration or FOCAL programming) would have resulted in only two to five times the speed of normal operation.

APPENDIX 4   PDP-8 PROGRAM

SYMBOLS AND ABSOLUTE LOCATIONS

ADADD	1106
ADCON	0161
ADDR	0023
ADEND	1107
ADST	1060
C	0100
CHANGE	0253
CONST	0026
CONT	0505
DACHN	0160
DACON	0146
DD	0125
DE	0130
DI	0145
DL	0117
DM	0553
DN1	0556
DN2	0561
END1	0024
ENTRY	0255
ER	0136
ERROR	0031
EX	0133
FINAL	0624
FIVE	0713
FLAG	1046
HUNDRD	0025
IL	0122
INCOST	0520
INDEX	0033
IT	0034
I1	0103
I2	0106
M	0564
MIDDLE	1223
M1	0020
ND	0141
NP	1342
NU	0144
NX	1345
N1	0111
N2	0114
ONE	1110
OUTPT	0600
PAUS	0032
P1	1356
P2	1361
READ	0214
RUN	1000
SET1	0400
SET2	0411
SET3	0426
SET4	0465
SND	1252
SNX	1200
SNXC	1237
START	0201
S1	1400
S2	1521
S3	1534
S4	1541
S5	1555
S6	1567
S7	1577
S8	1600
S9	1612
TEMP	0021
TEN	0716
THREE	0022
T1	1334
T2	1337
UPPER	1230
X1	1350
X1R	1331
X2	1353

EQUIVALENCE SPECIFICATIONS

```
FADD=1000
FSUB=2000
FMPY=3000
FDIV=4000
FGET=5000
FPUT=6000
FNOR=7000
FEXT=0
SQUARE=1
SQROOT=2
FSIN=3
FCOS=4
ARCTAN=5
FLOG=6
FEXP=7
FIX=10
NEGFAC=11
NEGOP=12
INPUT=13
OUTPUT=14
CRLF=15
NEWOUT=16
RNDFIX=17
FLOAT=4405
MSSGE=4406
IN=4407
FIXTAB
C=100
I1=103
I2=106
N1=111
N2=114
DL=117
IL=122
DD=125
DF=130
EX=133
ER=136
ND=141
X1R=1331
T1=1334
T2=1337
NP=1342
NX=1345
X1=1350
X2=1353
P1=1356
P2=1361
DM=553
DN1=556
DN2=561
M=564
```

PROGRAM LISTING

```
                *5
0005  4552            4552          /FLOAT SUBROUTINE ENTRY
0006  4400            4400          /MESSAGE PRINTOUT ENTRY
0007  5600            5600          /FLOATING POINT ENTRY
                *20                 /GENERAL QUANTITIES
0020  7777  M1,       -1
0021  0000  TEMP,     0
0022  0003  THREE,    3
0023  0100  ADDR,     100
0024  7637  END1,     -141
0025  0100  HUNDRD,   100
0026  0010  CONST,    0010
0027  3146            3146
0030  3147            3147
0031  7730  ERROR,    -50
0032  4000  PAUS,     4000
0033  0000  INDEX,    0
0034  0000  IT,       0
0035  0000            0
0036  0000            0
                *55
0055  7777            7777
0056  7777            7777
                *144
0144  0000  NU,       0
0145  0000  DI,       0
0146  0000  DACON,    0             /DIGITAL TO ANALOG
0147  6462            6462
0150  6464            6464
0151  7200            CLA
0152  1160            TAD DACHN
0153  6451            6451
0154  6454            6454
0155  6441            6441
0156  5155            JMP .-1
0157  5546            JMP I DACON
0160  0000  DACHN,    0             /DAC CHANNEL
0161  0000  ADCON,    0             /ANALOG TO DIGITAL
0162  6412            6412
0163  6421            6421
0164  6433            6433
0165  6411            6411
0166  5165            JMP .-1
0167  6421            6421
0170  6422            6422
0171  6411            6411
0172  5171            JMP .-1
0173  6421            6421
0174  7200            CLA
0175  6434            6434
0176  5561            JMP I ADCON
```

```
                *200
0200  7402              HLT
0201  6046      START,  TLS             /START OF MAIN PROGRAM
0202  4406              MSSGE           /INPUT MESSAGE
0203  1400              S1
0204  6257              -S2
0205  7200              CLA
0206  3033              DCA INDEX
0207  3034              DCA IT
0210  3035              DCA IT+1
0211  3036              DCA IT+2
0212  1025              TAD HUNDRD      /INPUT SEQUENCE
0213  3023              DCA ADDR
0214  4407      READ,   IN
0215  0013              INPUT
0216  6423              FPUT I ADDR
0217  0000              FEXT
0220  3021              DCA TEMP
0221  1022              TAD THREE
0222  1023              TAD ADDR
0223  3023              DCA ADDR
0224  1023              TAD ADDR
0225  1024              TAD END1
0226  7440              SZA
0227  5214              JMP READ
0230  7201              CLA IAC         /SET INITIAL STATES
0231  3160              DCA DACHN
0232  4407              IN
0233  5103              FGET I1
0234  3026              FMPY CONST
0235  0017              RNDFIX
0236  0000              FEXT
0237  7200              CLA
0240  1045              TAD 45
0241  4146              JMS DACON
0242  2160              ISZ DACHN
0243  4407              IN
0244  5106              FGET I2
0245  3026              FMPY CONST
0246  0017              RNDFIX
0247  0000              FEXT
0250  7200              CLA
0251  1045              TAD 45
0252  4146              JMS DACON
0253  7240      CHANGE, CLA CMA         /TOLERANCES CHANGED
0254  3144              DCA NU
0255  4777      ENTRY,  JMS RUN         /OPERATE THE SYSTEM
0256  4776              JMS SNX         /TERMINAL ERROR TEST
0257  4407              IN
0260  5775              FGET NX
0261  2133              FSUB EX
0262  0000              FEXT
0263  1045              TAD 45
0264  7710              SPA CLA
```

```
0265  5774            JMP OUTPT
0266  4773            JMS SND          /TERMINAL MATCHING TEST
0267  4407            IN
0270  5141            FGET ND
0271  2136            FSUB ER
0272  0000            FEXT
0273  1045            TAD 45
0274  7710            SPA CLA
0275  5772            JMP INCOST
0276  1144            TAD NU           /ITERATION SCHEME DECISIONS
0277  7450            SNA
0300  5771            JMP SET2
0301  7510            SPA
0302  5770            JMP SET1
0303  1020            TAD M1
0304  7550            SPA SNA
0305  5767            JMP SET3
0306  5766            JMP SET4
0366  0465
0367  0426
0370  0400
0371  0411
0372  0520
0373  1252
0374  0600
0375  1345
0376  1200
0377  1000
            PAGE
0400  4407  SET1,     IN               /FIRST COSTATE STEP
0401  5111            FGET N1
0402  1130            FADD DE
0403  6111            FPUT N1
0404  5141            FGET ND
0405  6353            FPUT DM
0406  0000            FEXT
0407  3144            DCA NU
0410  5777            JMP ENTRY
0411  4407  SET2,     IN               /SECOND COSTATE STEP
0412  5111            FGET N1
0413  2130            FSUB DE
0414  6111            FPUT N1
0415  5114            FGET N2
0416  1130            FADD DE
0417  6114            FPUT N2
0420  5141            FGET ND
0421  2353            FSUB DM
0422  6356            FPUT DN1
0423  0000            FEXT
0424  2144            ISZ NU
0425  5777            JMP ENTRY
0426  4407  SET3,     IN               /STEEPEST DESCENT STEP
0427  5114            FGET N2
0430  2130            FSUB DE
```

```
0431  6114           FPUT N2
0432  5141           FGET ND
0433  2353           FSUB DM
0434  6361           FPUT DN2
0435  5356           FGET DN1
0436  0001           SQUARE
0437  6364           FPUT M
0440  5361           FGET DN2
0441  0001           SQUARE
0442  1364           FADD M
0443  0002           SQROOT
0444  4130           FDIV DE
0445  6364           FPUT M
0446  5356           FGET DN1
0447  4364           FDIV M
0450  6356           FPUT DN1
0451  5361           FGET DN2
0452  4364           FDIV M
0453  6361           FPUT DN2
0454  5111           FGET N1
0455  2356           FSUB DN1
0456  6111           FPUT N1
0457  5114           FGET N2
0460  2361           FSUB DN2
0461  6114           FPUT N2
0462  0000           FEXT
0463  2144           ISZ NU
0464  5777           JMP ENTRY
0465  4407   SET4,  IN                 /REPEATED STEEPEST DESCENT
0466  5141           FGET ND
0467  2353           FSUB DM
0470  0000           FEXT
0471  1045           TAD 45
0472  7710           SPA CLA
0473  5305           JMP CONT
0474  4407           IN                 /REVERSE LAST STEP
0475  5111           FGET N1
0476  1356           FADD DN1
0477  6111           FPUT N1
0500  5114           FGET N2
0501  1361           FADD DN2
0502  6114           FPUT N2
0503  0000           FEXT
0504  5776           JMP CHANGE
0505  4407   CONT,   IN                 /CONTINUE REPEATED STEPS
0506  5111           FGET N1
0507  2356           FSUB DN1
0510  6111           FPUT N1
0511  5114           FGET N2
0512  2361           FSUB DN2
0513  6114           FPUT N2
0514  5141           FGET ND
0515  6353           FPUT DM
0516  0000           FEXT
```

```
0517  5777            JMP ENTRY
0520  4407  INCOST,   IN                   /TERMINAL CONDITION MET.
                                           /COST INCREASED
0521  5100            FGET C
0522  1125            FADD DD
0523  6100            FPUT C
0524  0000            FEXT
0525  5776            JMP CHANGE
0576  0253
0577  0255
            PAGE
0600  7200  OUTPT,    CLA
0601  1033            TAD INDEX
0602  7440            SZA
0603  5224            JMP FINAL
0604  2033            ISZ INDEX
0605  4407            IN                   /CHANGE TOLERANCES
0606  5125            FGET DD
0607  4313            FDIV FIVE
0610  6125            FPUT DD
0611  5130            FGET DE
0612  4313            FDIV FIVE
0613  6130            FPUT DE
0614  5133            FGET EX
0615  4316            FDIV TEN
0616  6133            FPUT EX
0617  5136            FGET ER
0620  4313            FDIV FIVE
0621  6136            FPUT ER
0622  0000            FEXT
0623  5777            JMP CHANGE
0624  7200  FINAL,    CLA
0625  1376            TAD (10
0626  3062            DCA 62
0627  1375            TAD (4
0630  3063            DCA 63
0631  6046            TLS                  /OUTPUT FINAL STATES
0632  4406            MSSGE
0633  1521            S2
0634  6244            -S3
0635  4407            IN
0636  5774            FGET X1R
0637  0014            OUTPUT
0640  0000            FEXT
0641  4406            MSSGE
0642  1534            S3
0643  6237            -S4
0644  4407            IN
0645  5773            FGET X2
0646  0014            OUTPUT
0647  0000            FEXT
0650  4406            MSSGE
0651  1541            S4
0652  6223            -S5
```

```
0653  4407            IN
0654  5111            FGET N1
0655  0014            OUTPUT
0656  0000            FEXT
0657  4406            MSSGE
0660  1534            S3
0661  6237            -S4
0662  4407            IN
0663  5114            FGET N2
0664  0014            OUTPUT
0665  0000            FEXT
0666  4406            MSSGE
0667  1555            S5
0670  6211            -S6
0671  4407            IN
0672  5100            FGET C
0673  0014            OUTPUT
0674  0000            FEXT
0675  4406            MSSGE
0676  1567            S6
0677  6201            -S7
0700  7200            CLA
0701  3063            DCA 63
0702  2062            ISZ 62
0703  4407            IN
0704  5034            FGET 1T
0705  0014            OUTPUT
0706  0000            FEXT
0707  4406            MSSGE
0710  1600            S8
0711  6166            -S9
0712  5772            JMP START
0713  0003    FIVE,   3
0714  2400            2400
0715  0000            0
0716  0004    TEN,    4
0717  2400            2400
0720  0000            0
0772  0201
0773  1353
0774  1331
0775  0004
0776  0010
0777  0253
              PAGE                        /OPERATE THE SYSTEM
1000  0000    RUN,    0                   /SET COST
1001  7200            CLA
1002  3160            DCA DACHN
1003  4407            IN
1004  5100            FGET C
1005  3026            FMPY CONST
1006  0017            RNDFIX
1007  0000            FEXT
```

```
1010  7200      .      CLA
1011  1045             TAD 45
1012  4146             JMS DACON
1013  2160             ISZ DACHN        /SET INITIAL COSTATES
1014  2160             ISZ DACHN
1015  2160             ISZ DACHN
1016  4407             IN
1017  5111             FGET N1
1020  3026             FMPY CONST
1021  0017             RNDFIX
1022  0000             FEXT
1023  7200             CLA
1024  1045             TAD 45
1025  4146             JMS DACON
1026  2160             ISZ DACHN
1027  4407             IN
1030  5114             FGET N2
1031  3026             FMPY CONST
1032  0017             RNDFIX
1033  0000             FEXT
1034  7200             CLA
1035  1045             TAD 45
1036  4146             JMS DACON
1037  6322             6322             /INITIAL CONDITION MODE
1040  7200             CLA              /PAUSE, CHARGE CAPACITORS
1041  1032             TAD PAUS
1042  3023             DCA ADDR
1043  2023             ISZ ADDR
1044  5243             JMP .-1
1045  6321             6321             /OPERATE MODE
1046  7200     FLAG,   CLA              /TEST RUN-END CONDITION
1047  4161             JMS ADCON
1050  1031             TAD ERROR
1051  7510             SPA
1052  5246             JMP FLAG
1053  6324             6324             /HOLD MODE
1054  7200             CLA              /ADC TERMINAL VALUES
1055  3160             DCA DACHN
1056  1306             TAD ADADD
1057  3023             DCA ADDR
1060  2160     ADST,   ISZ DACHN
1061  1160             TAD DACHN
1062  4161             JMS ADCON
1063  4405             FLOAT
1064  4407             IN
1065  4026             FDIV CONST
1066  6423             FPUT I ADDR
1067  0000             FEXT
1070  7200             CLA
1071  1023             TAD ADDR
1072  1022             TAD THREE
1073  3023             DCA ADDR
1074  1023             TAD ADDR
1075  1307             TAD ADEND
```

```
1076  7640            SZA CLA
1077  5260            JMP ADST
1100  4407            IN
1101  5034            FGET IT
1102  1310            FADD ONE
1103  6034            FPUT IT
1104  0000            FEXT
1105  5600            JMP I RUN
1106  1350  ADADD,    1350
1107  6414  ADEND,    -1364
1110  0001  ONE,      1
1111  2000            2000
1112  0000            0
            PAGE
1200  0000  SNX,      0                 /FIND STATE ERROR NORM
1201  4407            IN
1202  5350            FGET X1
1203  6331            FPUT X1R
1204  5350            FGET X1
1205  2117            FSUB DL
1206  6334            FPUT T1
1207  0000            FEXT
1210  1335            TAD T1+1
1211  7500            SMA
1212  5230            JMP UPPER
1213  4407            IN
1214  5350            FGET X1
1215  2122            FSUB IL
1216  6334            FPUT T1
1217  0000            FEXT
1220  1335            TAD T1+1
1221  7510            SPA
1222  5230            JMP UPPER
1223  7200  MIDDLE,   CLA
1224  3350            DCA X1
1225  3351            DCA X1+1
1226  3352            DCA X1+2
1227  5237            JMP SNXC
1230  7200  UPPER,    CLA
1231  1334            TAD T1
1232  3350            DCA X1
1233  1335            TAD T1+1
1234  3351            DCA X1+1
1235  1336            TAD T1+2
1236  3352            DCA X1+2
1237  4407  SNXC,     IN
1240  5350            FGET X1
1241  0001            SQUARE
1242  6345            FPUT NX
1243  5353            FGET X2
1244  0001            SQUARE
1245  1345            FADD NX
1246  0002            SQROOT
1247  6345            FPUT NX
```

```
1250  0000             FEXT
1251  5600             JMP I SNX
1252  0000  SND,       0                  /FIND TERMINAL ERROR NORM
1253  4407             IN
1254  5356             FGET P1
1255  0001             SQUARE
1256  6342             FPUT NP
1257  5361             FGET P2
1260  0001             SQUARE
1261  1342             FADD NP
1262  0002             SQROOT
1263  6342             FPUT NP
1264  5350             FGET X1
1265  4345             FDIV NX
1266  6334             FPUT T1
1267  5356             FGET P1
1270  4342             FDIV NP
1271  2334             FSUB T1
1272  6334             FPUT T1
1273  5353             FGET X2
1274  4345             FDIV NX
1275  6337             FPUT T2
1276  5361             FGET P2
1277  4342             FDIV NP
1300  2337             FSUB T2
1301  6337             FPUT T2
1302  5337             FGET T2
1303  0001             SQUARE
1304  6141             FPUT ND
1305  5334             FGET T1
1306  0001             SQUARE
1307  1141             FADD ND
1310  0002             SQROOT
1311  6141             FPUT ND
1312  0000             FEXT
1313  5652             JMP I SND

            PAGE
1400  0000  S1,        0
1401  3434  TEXT Z\\
1402  1031  HY
1403  0222  BR
1404  1104  ID
1405  4003   C
1406  1715  OM
1407  2025  PU
1410  2405  TE
1411  2240  R
1412  2317  SO
1413  1425  LU
1414  2411  TI
1415  1716  ON
1416  3434  \\
1417  3411  \I
1420  1611  NI
1421  2411  TI
1422  0114  AL
1423  7240  :
1424  4003   C
1425  1723  OS
1426  2454  T,
1427  4023   S
1430  2401  TA
1431  2405  TE
1432  2354  S,
1433  4003   C
1434  1723  OS
1435  2401  TA
1436  2405  TE
1437  2334  S\
```

```
1440  0000  Z
1441  2001  TEXT ZPA
1442  2201  RA
1443  1505  ME
1444  2405  TE
1445  2223  RS
1446  7240  :
1447  4022   R
1450  1107  IG
1451  1024  HT
1452  5440  ,
1453  1405  LE
1454  0624  FT
1455  4024   T
1456  0122  AR
1457  0705  GE
1460  2440  T
1461  2017  PO
1462  1116  IN
1463  2423  TS
1464  5440  ,
1465  0317  CO
1466  2324  ST
1467  4046   &
1470  4003   C
1471  1723  OS
1472  2401  TA
1473  2405  TE
1474  4023   S
1475  2405  TE
1476  2023  PS
1477  0000  Z
1500  3405  TEXT Z\E
1501  2222  RR
1502  1722  OR
1503  4016   N
1504  1722  OR
1505  1523  MS
1506  7240  :
1507  4023   S
1510  2401  TA
1511  2405  TE
1512  4046   &
1513  4024   T
1514  0522  ER
1515  1511  MI
1516  1601  NA
1517  1434  L\
1520  3400  \Z
1521  0000  S2,      0
1522  3434  TEXT Z\\
1523  0611  FI
1524  1601  NA
1525  1472  L:
1526  4040
1527  2324  ST
1530  0124  AT
1531  0523  ES
1532  4040
1533  0000  Z
1534  0000  S3,      0
1535  4040  TEXT Z
1536  0116  AN
1537  0440  D
1540  4000   Z
1541  0000  S4,      0
1542  3440  TEXT Z\
1543  4040
1544  4040
1545  4040
1546  4040
1547  0317  CO
1550  2324  ST
1551  0124  AT
1552  0523  ES
1553  4040
1554  0000  Z
1555  0000  S5,      0
1556  5634  TEXT Z.\
1557  4040
1560  4040
1561  4040
1562  4040
1563  0317  CO
1564  2324  ST
1565  4040
1566  0000  Z
1567  0000  S6,      0
1570  4040  TEXT Z
1571  5023  (S
1572  0503  EC
1573  1716  ON
1574  0423  DS
1575  5156  ).
1576  3400  \Z
1577  0000  S7,      0
            PAGE
1600  0000  S8,      0
1601  4040  TEXT Z
1602  1124  IT
1603  0522  ER
1604  0124  AT
1605  1117  IO
1606  1623  NS
1607  5634  .\
1610  3434  \\
1611  0000  Z
1612  0000  S9,      0
```

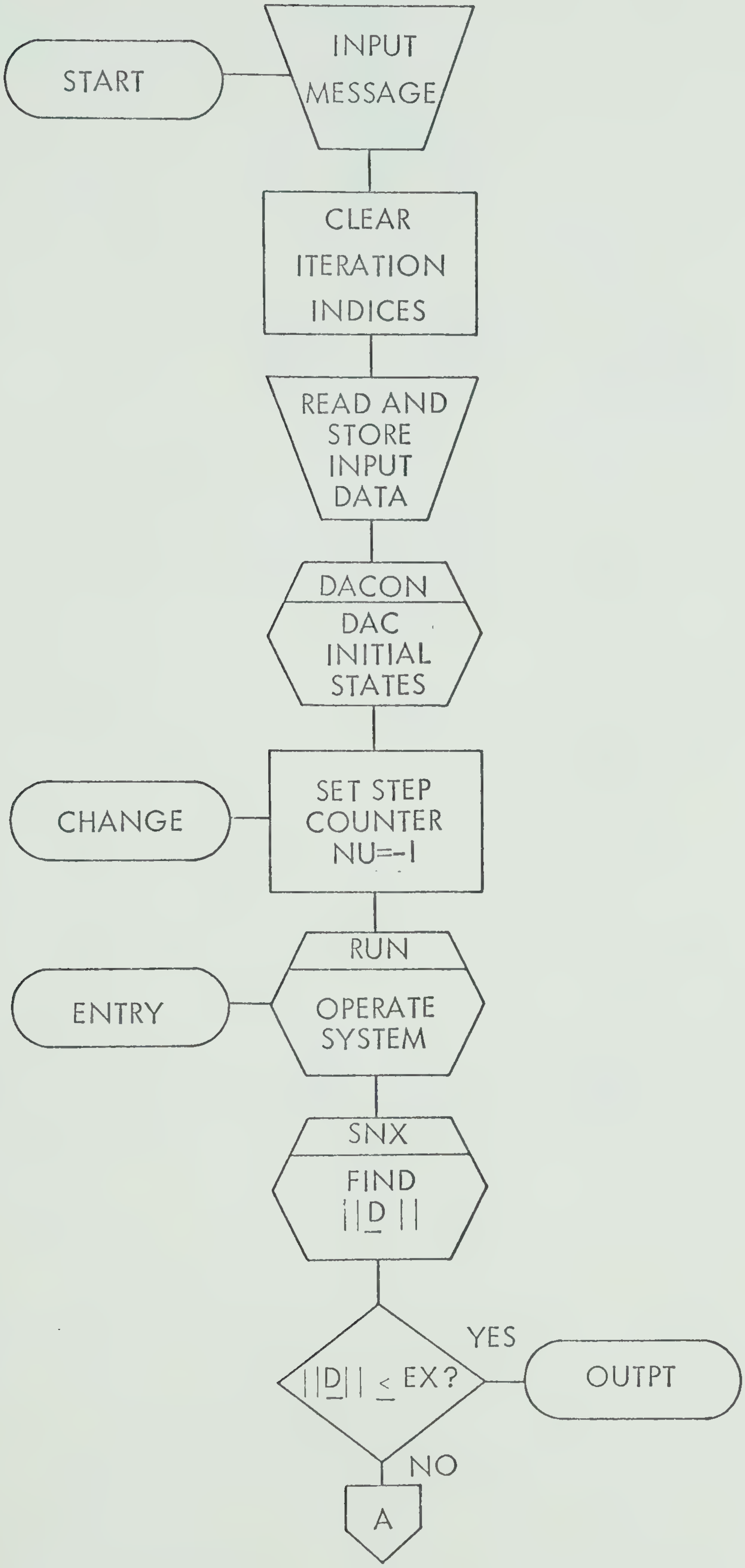
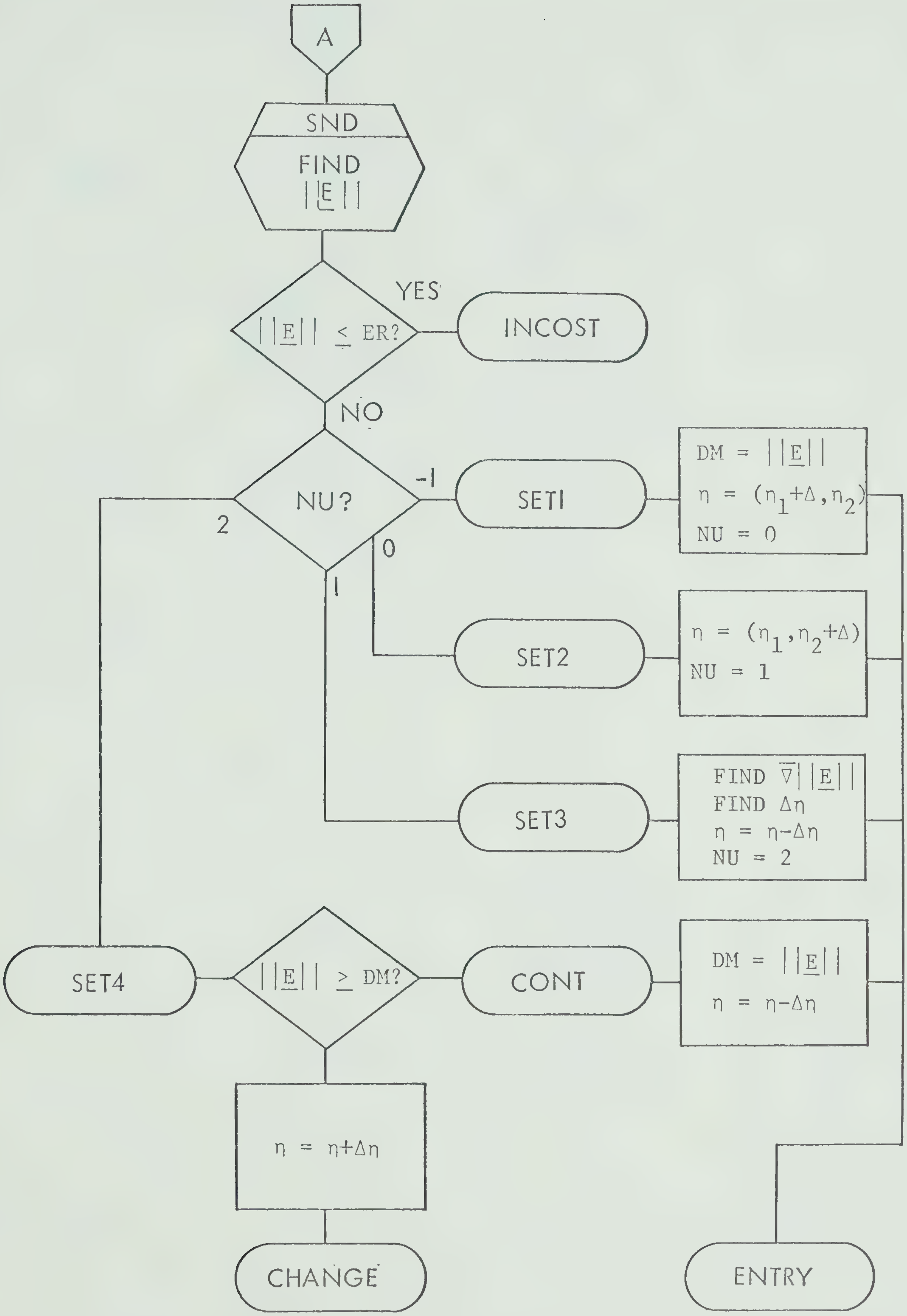
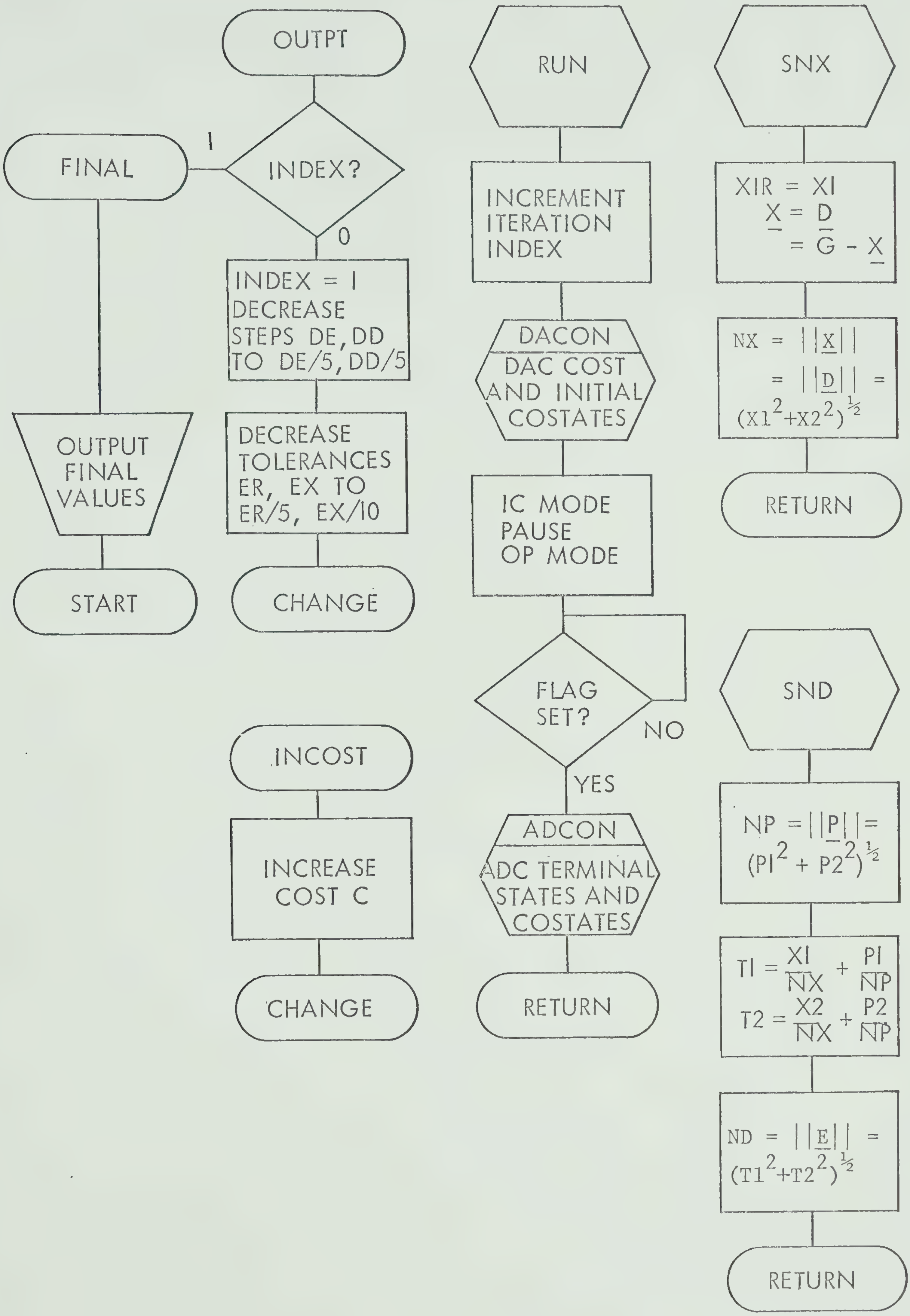

START
INPUT MESSAGE
CLEAR ITERATION INDICES
READ AND STORE INPUT DATA
DACON
DAC INITIAL STATES
CHANGE
SET STEP COUNTER NU=-1
RUN
ENTRY
OPERATE SYSTEM
SNX
FIND ||D||
||D|| ≤ EX?
YES
OUTPT
NO
A

FIGURE A - 4.1


A
SND
FIND
||E||
YES
||E|| ≤ ER?
INCOST
NO
NU?
-1
2
0
1
SET1
DM = ||E||
η = (η1+Δ, η2)
NU = 0
SET2
η = (η1, η2+Δ)
NU = 1
SET3
FIND ∇||E||
FIND Δη
η = η−Δη
NU = 2
SET4
||E|| ≥ DM?
CONT
DM = ||E||
η = η−Δη
η = η+Δη
CHANGE
ENTRY



OUTPT
INDEX?
1
0
FINAL
INDEX = 1 DECREASE STEPS DE, DD TO DE/5, DD/5
DECREASE TOLERANCES ER, EX TO ER/5, EX/10
OUTPUT FINAL VALUES
START
CHANGE
INCOST
INCREASE COST C
CHANGE
RUN
INCREMENT ITERATION INDEX
DACON
DAC COST AND INITIAL COSTATES
IC MODE PAUSE OP MODE
FLAG SET?
NO
YES
ADCON
ADC TERMINAL STATES AND COSTATES
RETURN
SNX
XIR = XI
X = D
= G - X
NX = ||X|| = ||D|| = (X1²+X2²)½
RETURN
SND
NP = ||P|| = (P1² + P2²)½
T1 = X1/NX + P1/NP
T2 = X2/NX + P2/NP
ND = ||E|| = (T1²+T2²)½
RETURN